NOVA IMS Information Management School

MGI

Mestrado em Gestão de Informação
Master Program in Information Management

Credit Scoring: Comparison of Non-Parametric Techniques against Logistic Regression

Miguel Mendes Amaro

Dissertation presented as partial requirement for obtaining the Master's degree in Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

CREDIT SCORING: COMPARISON OF NON-PARAMETRIC TECHNIQUES AGAINST LOGISTIC REGRESSION

by

Miguel Mendes Amaro

Dissertation presented as partial requirement for obtaining the Master's degree in Information Management, with a specialization in Knowledge Management and Business Intelligence

Advisor: Prof. Roberto Henriques, PhD

February 2020

ABSTRACT

Over the past decades, financial institutions have been giving increased importance to credit risk management as a critical tool to control their profitability. More than ever, it became crucial for these institutions to be able to well discriminate between good and bad clients for only accepting the credit applications that are not likely to default. To calculate the probability of default of a particular client, most financial institutions have credit scoring models based on parametric techniques. Logistic regression is the current industry standard technique in credit scoring models, and it is one of the techniques under study in this dissertation. Although it is regarded as a robust and intuitive technique, it is still not free from several critics towards the model assumptions it takes that can compromise its predictions. This dissertation intends to evaluate the gains in performance resulting from using more modern non-parametric techniques instead of logistic regression, performing a model comparison over four different real-life credit datasets. Specifically, the techniques compared against logistic regression in this study consist of two single classifiers (decision tree and SVM with RBF kernel) and two ensemble methods (random forest and stacking with cross-validation). The literature review demonstrates that heterogeneous ensemble approaches have a weaker presence in credit scoring studies and, because of that, stacking with cross-validation was considered in this study. The results demonstrate that logistic regression outperforms the decision tree classifier, has similar performance in relation to SVM and slightly underperforms both ensemble approaches in similar extents.

KEYWORDS

Credit scoring; models comparison; logistic regression; non-parametric models; machine learning; ensemble methods; heterogeneous ensemble

INDEX

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

Credit Scoring Model – model that expresses the probability of default of a potential borrower according to its characteristics. As a risk management instrument, banks and credit companies use them to protect them from lending money to clients that are not going to pay it back.

Parametric Methods – statistic methods where is assumed that data comes from a population that follows a distribution based on a fixed set of parameters.

Non-Parametric Methods – statistic methods that do not rely on the assumption of a fixed set of parameters distribution. Although it also relies on parameters, it is not a fixed set of them, and they are not defined in advance.

"Good" Client – desired client for the bank or credit company that is granting a credit. Client that respects the instalments' payments.

"Bad" Client – undesired client for the company. Client that does not respect its credit payment obligations, leading to financial losses for the banks or credit companies that lend the money.

Dependent or Target Variable – variable that we want to predict. In credit scoring problems, we want to predict if the client is going to default or not.

Independent or Predictor Variables – explanatory variables used to predict the dependent variable.

1. INTRODUCTION

1.1. BACKGROUND AND PROBLEM IDENTIFICATION

Banks and other financial institutions have been giving increased importance to risk management over the last decades. As such, many efforts are made to find more and better instruments to help those institutions in well identifying the several risks and being able to reduce their exposure to it. In particular, credit risk is one of the most significant risks that banks face (Cibulskienė & Rumbauskaitė, 2012). Specialized credit companies and banks, bound to several international regulations (from Basel Committee, for instance) and concerned about being competitive and profitable, are continuously searching for the most appropriate resources and techniques to manage their risk. Moreover, they need the best instruments to accurately tell the probability of a potential client to not pay his loan, at the moment he applies to it. That probability is commonly denominated as the probability of default. There is not a single definition of default since each institution defines its own concept according to its reality and aligned with the specific credit risk problem they are trying to manage. An example of a standard definition of default is the one in the Basel II framework which defines it as being "past due for more than 90 days on any material credit obligation to the banking group" (Basel Committee on Banking Supervision, 2006).

Credit scoring models are instruments used to come up with those probabilities, and they have been gaining popularity over the last decades. Being an instrument that was first used in the 1960s, it can briefly be defined as the use of statistical models to transform relevant data into numerical measures that will guide the credit decision (Anderson, 2007). In order to build a credit scoring instrument, analysts use historical data on the performance of already financed loans to understand what borrowers characteristics are determinant and useful for predicting whether the client will or not default (Mester, 1997). These models have evolved in a significant way over the last years, and in a general overview, they use techniques that allow us to segment them in two groups according to their statistical approach: parametric and non-parametric. Parametric models make assumptions regarding the data they use, such as normal distribution, linearity, homoscedasticity, and independence. On the other hand, the non-parametric models make no assumptions (or at least few). Logistic regression technique belongs to the first group (parametric models). In 2007, (Anderson, 2007) mentioned this technique as the most widely used in the field of credit scoring and (Lessmann, Baesens, Seow, & Thomas, 2015) in their benchmark study still pointed logistic regression as the industry standard technique. Critics exist towards this technique and, they mainly reside in the assumptions it makes and the violation of those same assumptions which can sacrifice the model accuracy.

As an alternative, non-parametric models are becoming more popular aligned with the progress in machine learning which, according to Arthur Samuel in 1959, it consists in giving the "computers the ability to learn without being explicitly programmed". According to studies already performed in the area, these later techniques show superior performance, especially in modelling nonlinear patterns when compared to statistical techniques (Wang, Hao, Ma, & Jiang, 2011). Among the several machine learnings algorithms that currently exist, there is not one that is regarded as the most suitable for credit scoring purpose (Wang et al., 2011). Decision tree algorithms, in particular, have been gaining popularity with the machine learning revolution. This technique uses mathematical formulas like Gini index to find an attribute of the data as well as a threshold value for that attribute to make splits of the input space (Patil & Sareen, 2016). These actions aim to come up, in the end, with different buckets

defined by splitting rules and filled with similar instances in terms of their class. Applied to the context of credit scoring, this method will find the splitting rules that efficiently distinguish the bad from the good clients in terms of their probability of default. Decision trees have showed success in solving credit scoring classification problems in several studies (Lee, Chiu, Chou, & Lu, 2006; Ince & Aktan, 2009). Moreover, this technique has a comparative advantage regarding other non-parametric popular methods such as neural networks or support vector machine: its results are rational and easy to interpret. Despite being easy to interpret and already have proven success in risk classification problems, decision trees used in a format of a single classifier have also evidenced propensity to overlearning the training data compromising its generalization capability. This problem, designated as overfitting, seem to be less present when dealing with ensemble models.

Ensemble models are methods where the predictions of a set of single learners are combined. The different existing methods can be grouped in homogeneous ensembles, where a single type of learning algorithm is used to generate the several base agents, and heterogeneous ensembles where different learning algorithms are mixed (Chapman, 2014). Generically, recent literature has been demonstrating that these learning methods, also designated as multiple classifier systems, are superior to single predictors in modelling probability of default (Wang et al., 2011; Lessmann et al., 2015). However, heterogeneous ensembles have not been as frequently approached in credit scoring studies as other techniques mentioned earlier, which makes it more difficult to be certain about their advantages against other techniques such as logistic regression.

1.2. STUDY RELEVANCE AND MOTIVATIONS

Banks and other financial institutions, mostly using logistic regression or other parametric methods may not be taking full advantage of recent progress in machine learning and available non-parametric approaches. This dissertation has the main objective of quantifying the impacts in performance a financial institution can expect from switching its current logistic regression model to a more complex model developed using a non-parametric technique. In fact, this knowledge extension can be seen as a benefit not only for financial institutions, due to the fact that better predictive techniques can result in higher profits, but also for academics.

To achieve the given objective, this dissertation measures the performance of several modelling techniques against logistic regression. Several models' benchmarks were performed in previous literature and the set of techniques being compared in this particular dissertation reunites more studied predictive techniques with less present in previous works machine learning methods. More concretely, the techniques being compared against logistic regression consist of two popular single classifiers techniques, decision tree and SVM with RBF kernel, and two ensembles approaches. One homogeneous type (random forest) and one heterogeneous type of ensemble approach (stacking with cross-validation).

The relevance of this study to the literature is reinforced by the inclusion, in the pool of algorithms being compared, of a heterogeneous ensemble approach. Comparing to homogeneous ensembles, heterogeneous ensembles are much less frequent in credit scoring studies and that is an important factor for adding value to this dissertation. The heterogenous ensemble technique being used is StackingCVClassifier (from Mlxtend library), which consists of a renewed version of the original StackingClassifier with the difference that uses cross-validation technique to prepare the input data for the meta-classifier, and by doing that, has a stronger overfitting control.

Another relevant aspect of this study is the use of four different real-life credit datasets to be able to draw more consistent conclusions (one from Kaggle community and the others from UCI Machine Learning Repository). Moreover, the comparison is enriched with a set of different performance indicators and more adjusted model training and evaluation methods that have in consideration the class imbalance underlying the datasets being used. The models compared were tuned using an efficient hyperparameter tuning technique named RandomizedSearchCV from Scikit-learn library.

2. LITERATURE REVIEW

There is a considerable amount of literature that investigates the application of several statistical and machine learning techniques in the context of credit scoring. These models, given the importance they have for credit companies and banks on their risk management strategies, continue being subject of analysis in the literature.

As previously mentioned, logistic regression is still widely used for credit scoring purposes (Abdou, Pointon, & El-Masry, 2008; Hand & Henley, 1997). This method presents an added value when compared to linear regression, as it is more appropriate when we need binary outcomes. Computational developments made it more viable (Anderson, 2007), and financial institutions under Basel II felt a greater incentive to adopt it as estimated probability of default became needed. Its results are considered easy to interpret, and that became of increased importance when regulators started requiring banks to justify their credit application rejections (Qiu, Kandhai, & Sloot, 2010). However, logistic regression, like any other parametric technique, lives with underlying assumptions that, when violated, compromise the model results' credibility (Anderson, 2007). That scenario suggests that research should be conducted to evaluate the relevance of using an alternative approach such as non-parametric techniques once they are regarded as superior to statistical ones when modelling nonlinear features (Wang et al., 2011).

Classification and regression tree (CART) is an example of a non-parametric technique. In relation to logistic regression, it has the comparative advantage of being able to capture nonlinear relationships between input variables. Classification and regression tree models have showed success in solving credit scoring problems throughout the literature. A study from (Lee et al., 2006) performed credit scoring tasks using CART and MARS models against other techniques. Results have demonstrated that CART and MARS models outperformed discriminant analysis, logistic regression, neural networks and support vector machine by reaching a superior average classification rate. Additionally, an experiment from (Ince & Aktan, 2009) compared CART, neural networks, logistic regression and discriminant analysis techniques in terms of accuracy and CART demonstrated the highest average classification rate. On the other hand, error Type II was also measured, and, in that case, neural networks outperformed the other three models. This aspect is particularly relevant in credit scoring context given that this type of error results in higher misclassification costs when compared with Type I error.

Support vector machine is an example of another popular non-parametric technique. Despite being an accurate nonlinear approach, with historical data of large dimension, it can become very computationally expensive. In response to that, (Harris, 2015) assessed the performance of clustered support vector machine and obtained well-compared performances (in relation to support vector machines) in terms of accuracy combined with a relatively cheap computational effort. Along with neural networks, support vector machine algorithms are considered as “black box” models due to the lack of information regarding the relationships between variables (Qiu et al., 2010). CART model has an advantage regarding this issue as it produces results which are easy to interpret. In the particular context of credit scoring, this aspect remains critical as "black-box" models are seen with suspicion and scepticism in the banking sector (Khandani, Kim, & Lo, 2010). On the other hand, it appears to be consensual in the literature about applications of credit scoring modelling techniques that decision trees are prone to overfit. (Zakirov, Baxter, Bartlett, & Frean, 2015) (L, Natarajan, Keerthana, Chinmayi, & Lakshmi, 2016) (Bach, Zoroja, Jakoviü, & Šarlija, 2017).

The relatively recent machine learning revolution brought a renewed interest in finding new methods of classification and the ensemble methods have been drawing attention from researchers. These methods seem to help in controlling the problem of overfitting associated with the decision trees models. In fact, (Hu et al., 2006) performed an experimental comparison of several ensemble methods (BaggingC4.5, AdaBoostingC4.5, and random forest) against the C4.5 model. All ensemble methods outperformed the C4.5 single classifier, demonstrating a significantly higher accuracy. In more recent literature, (Chopra & Bhilare, 2018) also compared ensemble tree learning techniques with the single learner decision tree and the empirical analysis showed that the ensemble technique gradient boosting outperformed the single tree classifier.

Regarding ensemble methods, one can classify them in two groups, homogeneous and heterogeneous, depending on if these are formed using a single classification algorithm or if they mix several ones. Although homogeneous ensembles use a single classification algorithm, their base learners are trained using different parts of the dataset or using different feature sets. For example, bagging, boosting and staking are well-known ensemble approaches for modifying the training dataset on credit scoring problems (Wang, Hao, Ma, & Jiang, 2011). Interested in comparing these different approaches of using the dataset in ensemble methods, Dietterich (2000) studied the performance of three methods (Randomizing, Bagging and Boosting) and Boosting returned the best results in most of the tasks assessed. However, it was also evident that under noisy data environment Bagging was the best. Later, Caruana & Niculescu-Mizil (2006) performed an empirical comparison between ten supervised learning methods, in which the boosted trees also demonstrated to be the best learning algorithm overall, and the random forests (bagging method) were the second best. On the contrary, (Lessmann, Baesens, Seow, & Thomas), in 2015, performed an update to the study of (Baesens et al., 2003), in which both single and ensemble classifiers were compared and in the specific family of homogeneous ensembles, random forest had the strongest performance, being superior, among others, to boosted trees.

It is not uncommon to find articles demonstrating ensemble classifiers outperforming logistic regression. However, it is also not difficult to find in the literature that these models do not beneficiate of logistic regression's great advantage of being easy to interpret. As an example, (Hue, Hurlin, & Tokpavi, 2017) state that logistic regression underperforms ensemble models such as random forest due to its pitfalls in modelling nonlinear effects, but then, on the other hand, random forest prediction technique lacks parsimony, i.e. uses many parameters, which make the model less relevant in credit scoring applications where decision-makers need interpretable rules. Concerned with the subject of lacking interpretability to advanced machine learning techniques, these authors (Hue et al., 2017) studied the possibility of improving logistic regression model with nonlinear decision trees effects. They have discovered that the resulting model outperformed traditional logistic regression while being competitive with random forest achieving an efficient trade-off between performance and interpretability. In a similar line of thought, (Vanderheyden & Priestley, 2018), underline the importance of models developed in a regulated industry to be interpretable and rational. Once that is not always the scenario associated with machine learning algorithms, they have proposed a methodology that blends logistic regression with a core model enhancement strategy. In practice, it was created an ensemble that uses a linear combination of predictions from a set of logistic regression models. The fact of using a linear combination of predictions is referred to as the critical point to achieve the interpretability and rationality required in regulated industries.

While homogeneous ensemble methods mostly combine the predictions of the base learners by simple or weighted voting, the heterogeneous ensemble models present more complex methods to combine the base learners' predictions. One of them is stacking and (Dzeroski & Bernard, 2011), through comparing several approaches of it, have shown that this approach achieves improved performance gains in relation to pick the best classifier from the ensemble cross-validation. Regarding the comprehensive benchmark study performed by (Lessmann et al., 2015), heterogeneous ensembles have demonstrated higher performance, but still, it is a field yet to be explored in the literature as table 1 suggests. The table sums up the literature over the last ten years related with applications of ensemble methods in the credit scoring field, and it is clear that homogeneous ensembles have been more present in the several studies and heterogeneous ensemble more neglected. Likewise, table 1 reveals that most of the studies have used one or two datasets to draw conclusions of the modelling techniques which compromises the reliability and consistency of conclusions made.

Year	Study	# Datasets	Homogeneous Ensemble	Heterogeneous Ensemble
2010	Zhang et al.	1	x	
2010	Zhou et al.	2	x	
2010	Yu et al.	2	x	
2011	Finlay	2	x	
2011	Wang et al.	3	x	x
2012	Brown and Mues	5	x	
2012	Wang et al.	2	x	
2012	Li et al.	2	x	
2012	Wang and Ma	1	x	
2012	Marqués et al.	6	x	
2013	Kruppa et al.	1	x	
2014	Tsai	5	x	x
2014	Abellán and Mantas	3	x	
2015	Lessman et al.	8	x	x
2016	Zhou et al.	1	x	
2016	Kumar et al.	1	x	
2017	Bequé et Lessmann	3	x	
2018	Lawi et al.	2	x	
2018	Chopra et Bhilare	1	x	
2018	Vanderheyden et Priestley	1	x	
2018	Luo	1	x	
2019	Papouskova et Hajek	2	x	x
2019	Castro Vieira et al.	1	x	
Average / Counts		**2,4**	**23**	**4**

Table 1 - Overview of ensemble models applications in credit scoring literature

3. EXPERIMENTAL DESIGN

This chapter focus on describing the methodology carried in this study. Firstly, basic information concerning the four real-life datasets is presented. Then, follows an explanation concerning the methods used for data pre-processing and feature selection. Afterwards, there is a concise description of each of the algorithms being compared in this study. Specificities of the modelling work are detailed after as well as the indicators and analysis performed to enable the comparison of the several models developed across the four datasets.

3.1. CREDIT DATASETS

In this dissertation, four real-world credit datasets were used to train and test all the models. The "Give Me Some Credit" dataset referred in this dissertation as the Kaggle Dataset was the dataset used in of the competitions launched by the Kaggle community, a community of data scientists owned by Google, Inc. The other three datasets (referred in this dissertation as the Japanese, German and Australian datasets) are from the UCI Machine Learning Repository and were used several times in previous works (Baesens et al., 2003; Lessmann et al., 2015; Wang et al., 2011). All the four datasets reflect a typical credit scoring classification problem where the variable being predicted distinguishes between desired and undesired credit clients (non-default and default cases respectively). As can be observed in table 2, the Kaggle dataset is by far the largest one of the four, containing 150 thousand of records, whereas the other three have a thousand records maximum. In terms of balance between the two classes of the target variable (default and non-default), the Kaggle dataset is also the most imbalanced one (only 7% of default records), followed by the German Dataset (30% of default records) and then the Australian and Japanese dataset, that are not far from being perfectly balanced (56% of default records).

The features of the datasets combine social with behavioural information (as a client of a credit product) concerning each individual client. This characteristic is true for Kaggle and German datasets but not confirmed in Japanese and Australian dataset since these last two have their feature names and values codified to protect data confidentiality. All datasets except Kaggle dataset contain both numeric and categorical type of features.

	Kaggle Dataset	Japanese Dataset	German Dataset	Australian Dataset
Title	Give Me Some Credit	Credit Approval	German Credit Data	Australian Credit Approval
Source	Kaggle	UCI Repository	UCI Repository	UCI Repository
Number of Records	**150 000**	**690**	**1 000**	**690**
of which Default	10 026	383	300	383
of which Non-Default	139 974	307	700	307
% of Default	7%	56%	30%	56%
Number of Features	**10**	**15**	**20**	**14**
of which Numeric	10	6	7	6
of which Categorical	0	9	13	8

Table 2 - General description of the datasets

The four datasets can be found in the websites below:

https://www.kaggle.com/c/GiveMeSomeCredit/

https://archive.ics.uci.edu/ml/datasets/Japanese+Credit+Screening

https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

http://archive.ics.uci.edu/ml/datasets/statlog+(australian+credit+approval)

3.2. DATA PRE-PROCESSING

The models assessed in this study have different “needs” of data pre-processing, for example, logistic regression has a greater need for outliers' removal than the other models, and it is more prone to poor performance when correlated variables are used. However, a single pre-processing approach was performed for each one of the datasets, which means that all models were trained and evaluated using the same transformation of the original data. The decision was taken with the underlying belief that, with these assumptions, the results of the several models would be more comparable between them, which leads to more valid conclusions.

The pre-processing actions described in the following points translate the effort done to get the data in its best shape to feed the models.

3.2.1. Missing values

The Kaggle dataset and the Japanese dataset present, in their original form, features containing missing values (two features and seven features with missing values, respectively). Almost all features have a relatively low percentage of missing values (2.6% maximum). There is a feature that is an exception to this statement which is the "MonthlyIncome" feature of the Kaggle dataset, that contains missing values in almost 20% of the records and for that reason received special treatment.

Concerning the eight features with a low missing value rate, the action performed was:

- For numerical features: replace the missing values with the median of the non-missing value records
- For categorical features: replace the missing values with the most frequent non-missing value

The “MonthlyIncome” feature, on the other side, received a more customized treatment, having in consideration the age of the client (a feature that had no missing values). The clients were classified in eight age classes, and the median of the "MonthlyIncome" was calculated for each age class (figure 1). Finally, the missing values were replaced with the respective income according to the age class the client belongs to.

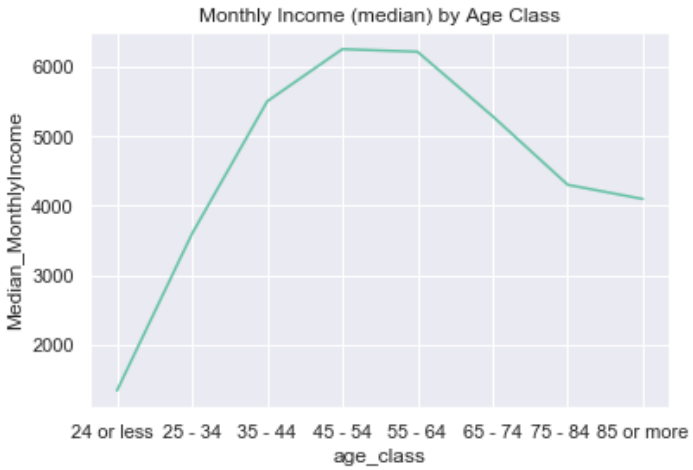

Figure 1 - Median monthly income calculated for each age class

3.2.2. Outliers and feature engineering

The outlier treatment was differentiated according to the variable's type and cardinality:

- **Numerical variables:**

A graphic analysis of the distribution was performed (figure 2). That analysis served as a basis for defining a cut-off point above (or/and below) which the observations were considered outliers and, therefore, replaced by the median (figure 3).

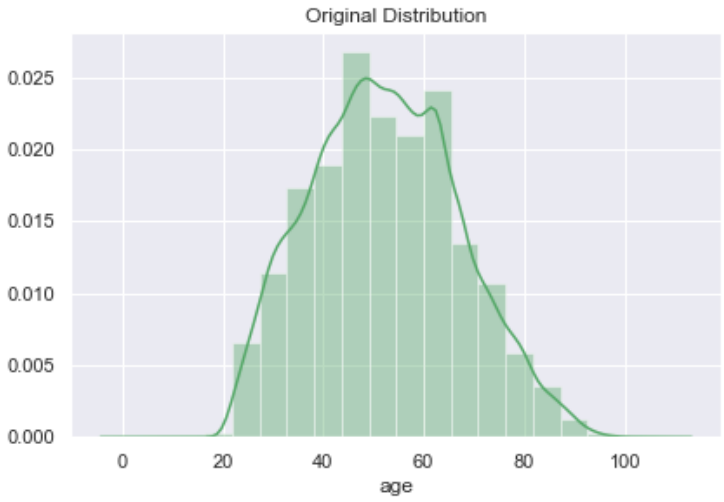

Figure 2 - Outlier treatment for numeric variables (before)

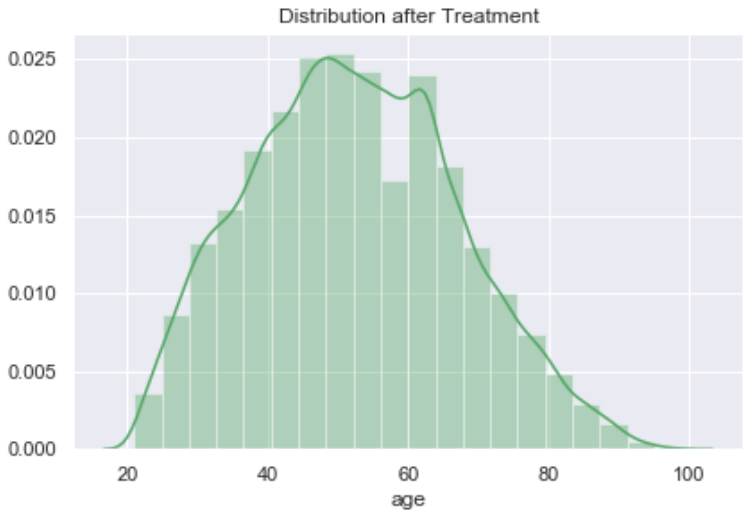

Figure 3 - Outlier treatment for numeric variables (after)

- **Categorical variables with a low number of categories**:

The outlier category was replaced by another, more frequent, category that showed to be the most similar one in terms of risk. To assess this similarity, the percentage of default records in each category was calculated (table 3). For outlier categories with extremely low frequency, the treatment was simpler, more specifically, these were replaced directly with the most frequent category.

Original Form		
Categories	% of Default Records	Total Number of Records
A141	41%	139
A142	40%	47
A143	28%	814
Total	**30%**	**1 000**

After Treatment		
Categories	% of Default Records	Total Number of Records
A141	41%	186
A143	28%	814
Total	**30%**	**1 000**

Table 3 - Outlier treatment (categorical variables with a low number of categories)

- **Categorical variables with a high number of categories:**

In this case, the variables were transformed grouping all the original categories into new and larger categories. This transformation was performed, having into consideration both risk level and frequency of each of the new categories. Moreover, it was attempted to create categories with not so different relative frequencies and the most different as possible in terms of risk, so they would be capable of discriminating default from non-default cases (table 4).

Original Categories	Count	% of Default Records	New Categorie
14	38	16%	R1
13	41	29%	R1
12	3	33%	R1
11	78	35%	R1
10	25	44%	R2
9	64	48%	R2
8	146	55%	R3
7	38	58%	R3
6	54	65%	R4
5	10	70%	R4
4	51	73%	R4
3	59	76%	R5
2	30	77%	R5
1	53	87%	R5
Total	690	56%	

New Categorie	Count	% of Total	% of Default Records
R1	160	23%	29%
R2	89	13%	47%
R3	184	27%	55%
R4	115	17%	69%
R5	142	21%	80%
Total	690	100%	56%

Table 4 - Outlier treatment (categorical variables with a high number of categories)

Besides the feature transformation described in the previous point, all the categorical variables were transformed into numerical ones. This was mandatory because not all algorithms would be capable of handling categorical features (for instance, the decision tree model would but logistic regression would not). One popular way to perform this transformation is to one-hot-encode the categorical features, especially when there is no natural order between the feature's categories (Ferreira, 2018). The result of this transformation is a dataset with a new feature for every single value assumed by the categorical features before the transformation. One could have used the OneHotEncoder function of Scikit-learn library, however since it requires the inputs to be integers and the Japanese and German datasets contain their categorical variables with string values, a more direct alternative was used. That alternative can handle strings as inputs, consists of a function with the same name as the one from Scikit-learn (OneHotEncoder) and can be found in Category Encoders library.

Since standardizing input data is needed for several machine learning algorithms (for example for support vector machines with RBF kernel), after one-hot-encoding the categorical features, all predictive features were centred and scaled using Scikit-learn StandardScaler function. In other words, all the features were transformed, having in the final form, mean equal to zero and standard deviation equal to one.

3.3. FEATURE SELECTION

Perform dimensionality reduction is an important step to avoid overfitting, especially when one is dealing with small datasets such as three of the four datasets used in this study (namely, the Japanese, the German and the Australian datasets). Apart from avoiding overfitting, there are other advantages in performing such a task. For instance, it can reduce the noise and redundancy brought into the model that can sacrifice its accuracy, it reduces the computing effort, and it turns the model simpler and, hence, more easily understandable.

Feature Selection was performed in three major steps: measure of feature importance, analysis of correlation, analysis of the optimal number of features assessing performance on the test set.

To measure feature importance, the ANOVA F-statistic was calculated. This metric can be used in classification problems, such as the one studied in this dissertation, to evaluate the relationship between the predictive features and the target variable. The calculation was achieved using the Scikit-learn feature selection function called f_classif (figure 4).

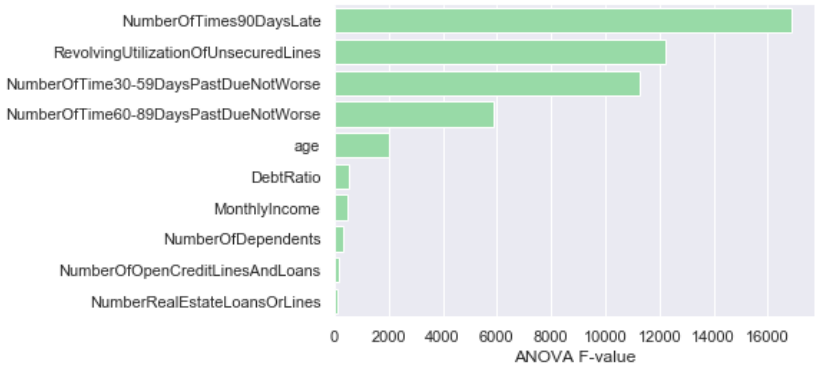

Figure 4 - Feature importance analysis (Kaggle dataset)

To assess the correlation between the predictive features, the Spearman's rank correlation coefficient was used (figure 5). This coefficient has an advantage in relation to Pearson's correlation coefficient in the way it measures not only linear but also nonlinear relations between variables. In every pair of variables with Spearman's rank correlation coefficient above 0.5, one of the variables was disregarded, more concretely, the one with lower ANOVA F-statistic value.

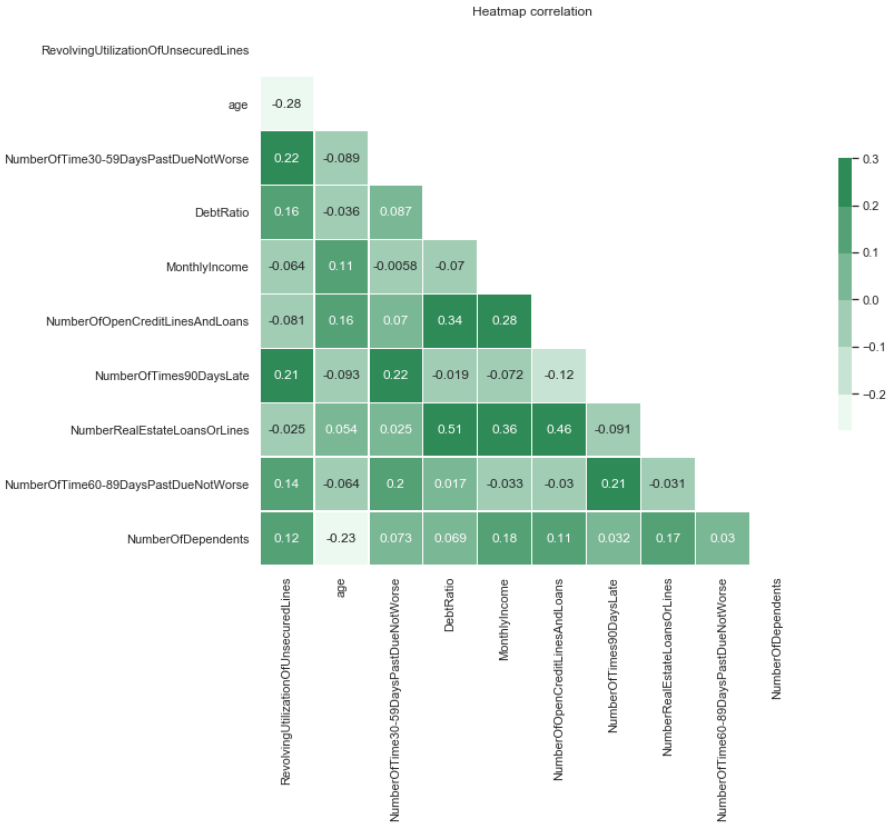

Figure 5 - Spearman's correlation coefficient (Kaggle dataset)

The two previous steps were complemented with a third analysis where the most important not strongly correlated variables were ordered and used to train a logistic regression model. That model was trained and tested in terms of performance (AUC metric) in several iterations (each iteration using a different number of features) to find the optimal number of features to be included in the models of each dataset. The cut-off point that was searched for in every dataset was the point where adding an additional variable to the logistic regression model had no longer significant gains in terms of performance measured on the test set (figure 6). The optimal number of features concerning each dataset is specified in table 5, and the final set of features included in the model is specified in table 6.

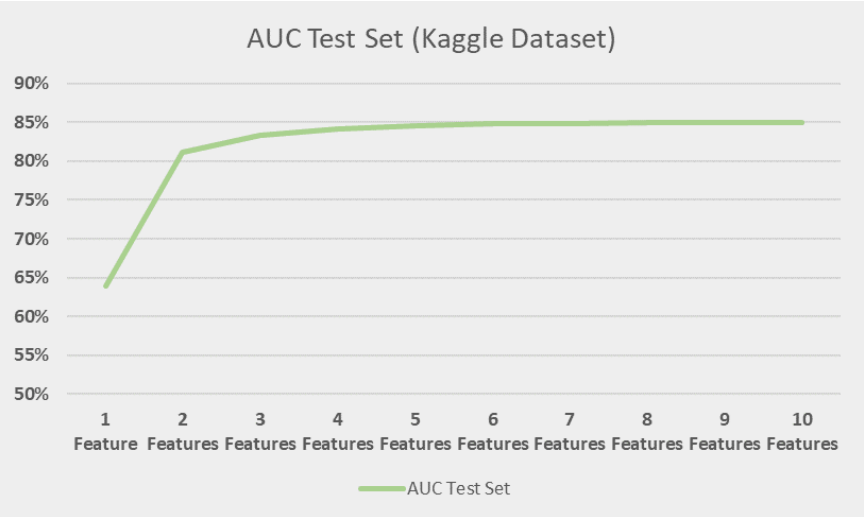

Figure 6 - AUC by number of features included in the model

	Kaggle Dataset	Japanese Dataset	German Dataset	Australian Dataset
Number of Features in the Original Dataset	**10**	**15**	**20**	**14**
Number of Features after One Hot Encode	**10**	**28**	**50**	**26**
Number of Features included in the models	**6**	**7**	**8**	**4**

Table 5 - Optimal number of features by dataset

Kaggle Dataset	Japanese Dataset	German Dataset	Australian Dataset
1. NumberOfTimes90DaysLate	1. A9_t	1. A1_A14	1. A8_t
2. RevolvingUtilizationOfUnsecuredLines	2. A10_t	2. A1_A11	2. A10
3. NumberOfTime30-59DaysPastDueNotWorse	3. A8	3. A2	3. A7
4. NumberOfTime60-89DaysPastDueNotWorse	4. A6_R1	4. A6_A61	4. A5_R1
5. Age	5. A6_R5	5. A3_A34	
6. DebtRatio	6. A7_R1	6. A3_A30	
	7. A4_u	7. A15_A152	
		8. A3_A31	

Table 6 - Final set of features included in the models

3.4. ALGORITHMS OVERVIEW

This section gives a general overview of the algorithms used in this study, starting with the three single learning approaches (logistic regression, support vector machines and decision trees) and followed by the two ensemble learning approaches (random forest and stacking with cross-validation). This last algorithm (stacking with cross-validation) is the only technique that was not retrieved from the Scikit-learn library (it was retrieved from Mlxtend library). Some key parameters of these algorithms are also

discussed as they were tuned during the modelling process as described in the Modelling technique section of this dissertation.

3.4.1. Single classifiers

3.4.1.1. Logistic regression

Logistic regression was developed by the statistician David Cox in 1958, and it consists of a parametric method that is widely used in the credit scoring models among financial institutions. The fact that logistic regression can be used to directly predict probabilities (Brid, 2018) is one of the main differences comparing to linear regression and it is quite useful when our dependent variable is categorical. In fact, linear regression has an output that can fall out of the range 0 to 1, being for that reason inappropriate for classification problems. Therefore, logistic regression is better suited for the particular context of credit scoring, given that we intend to predict the probability of a certain event to occur. Being more concrete, we want to predict the probability of a certain client to default (probability of our dependent variable being 1).

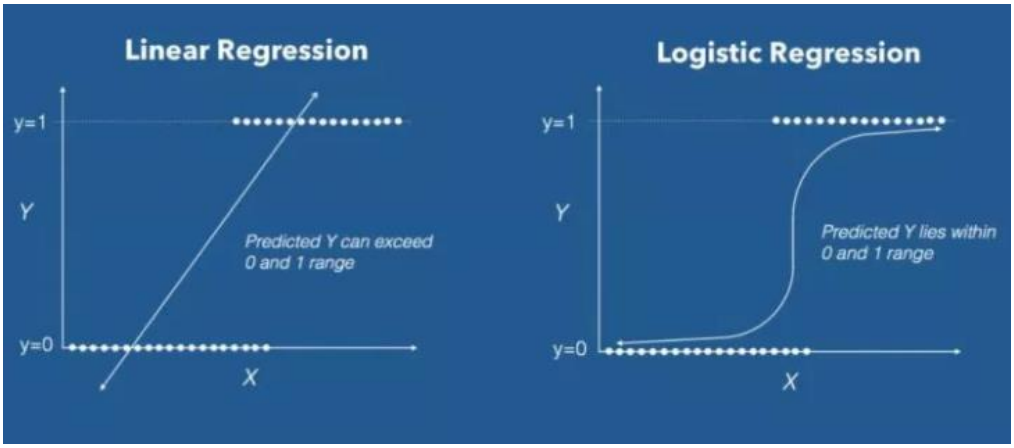

Figure 7 - Linear regression vs logistic regression

From: https://www.machinelearningplus.com/machine-learning/logistic-regression-tutorial-examples-r/

In logistic regression, a link function known as logit, the natural log of the odds, is used to establish a linear function with the input variables. In credit scoring context, with this link function that transforms our outcome variable, logistic regression manages to model a nonlinear association between the probability of default and the input variables in a linear way (formula 1). The logit function is also what enables to scale the independent variables to the scale of a probability (MacKenzie et al., 2018).

In the same way linear regression uses the method of ordinary least squares to estimate the coefficients that provide the best fitting, logistic regression uses a technique called Maximum Likelihood Estimation that makes a guess for the values of the coefficients and iteratively changes them in order to maximize the log of the odds (Anderson, 2007). In other words, during the fitting process, this method will estimate the coefficients in such a way it will maximize the probability of being default of the individuals labelled as default as well as maximize the probability of not being default of the individuals labelled as non-default.

In the end, it results in a regression formula with the following form:

$$\ln\left(\frac{p(default)}{1-p(default)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \varepsilon \quad (1)$$

where k is the number of independent variables.

That can be converted into a probability:

$$p(default|X) = \frac{e^{\beta_0 + \beta_1 x1 + \beta_2 x2 + \cdots + \beta_k xk}}{1 + e^{\beta_0 + \beta_1 x1 + \beta_2 x2 + \cdots + \beta_k xk}} \quad (2)$$

As logistic regression outcomes a probability, a cut-off probability or threshold needs to be fixed. As an example, if one fixes a threshold of 0.5 the model will classify every sample with a probability of default above 0.5 as positive (default) and every sample with a probability below 0.5 as negative (non-default).

In the past, there was a great disadvantage related to this model, which was its computational intensiveness, but with the progress in computer's capacities, this problem was overcome. That led to an increase in the popularity of this technique which is valued for being specifically designed to handle binary outcomes and for providing a robust estimate of the actual probability. Although logistic regression is still regarded as the industry standard technique (Lessmann et al., 2015), it is not free from several critics in literature that mainly reside in the assumptions it takes such as linearity between the independent variables and the log of the odds, or the absence of correlation among the predictors as well as the relevance of the predictors (Anderson, 2007). These assumptions are easily violated and naturally compromise the model's accuracy. The limited complexity associated with this technique is the main disadvantage in comparison to non-parametric more advanced machine learning techniques. Still, logistic regression has a strong advantage compared to other models which is the easy interpretation of its results. In fact, the coefficients associated to each independent variable that are estimated during the fitting process reveal, in a very direct way, the importance of each input variable as it expresses the size of their effect on the dependent variable.

The function available in Scikit-learn's library for using this model enables several parameters that can be tuned. Penalty is one of them, and it specifies the norm used in penalization. Another parameter used for model tuning is the C parameter that expresses, in an inverse way, the strength of the regularization.

3.4.1.2. Support vector machines

The interest in non-parametric techniques for credit scoring has been increasing. In opposition to parametric methods, they do not rely on many (or even any) assumptions about the data which release us from the risk of violating them during the modelling process and having losses of performance in our final predictive model. Machine learning and its increasingly obvious positive effects on business management have been supporting a renewed interest on relatively recent machine learning algorithms (Hue et al., 2017).

Support vector machines are strong machine learning models used in supervised learning both for classification and regression problems. The original form was firstly invented by Vladimir Vapnik and Alexey Chervonenkis in 1963 with the construction of a linear classifier. Later, (Boser, Guyon, & Vapnik,

1992), presented an SVM technique applicable to nonlinear classification functions such as polynomial and radial basis function stating good generalization when compared to other learning algorithms on optical character recognition problems. The approach followed by these authors consisted of applying the kernel trick to find the hyperplane that maximizes the margin. To understand how the support vector machines work, it is important for one to get familiar with the concept of margin. The margin is the distance between the decision line of the classifier and the closest observation. In the simplest scenarios, SVM models search for the maximal margin in the input space, which represents, with respect to binary classification problems, the largest buffer separating observations of the two different classes (Kirchner & Signorino, 2018). Depending on the number of predictive variables, the separating boundary can be a line if it has two predictive variables, a plane if it has three predictive variables or a hyperplane if it has four or more variables. The left panel of figure 8 (Kirchner & Signorino, 2018) exhibits an unrealistic scenario in terms of simplicity in which data is completely linearly separable. The input space has two predictive variables, X_1 and X_2, and the estimated plane H separates successfully any observation with no misclassification errors. In fact, in a scenario such as the one presented on the left panel of figure 8 there is an infinite number of hyperplanes that are capable of separating the observations of the two classes, but one natural choice is the maximum margin classifier which consists of the separating hyperplane that is farthest from the training observations (James, Witten, Hastie, & Tibshirani, 2013). The especially signalled triangles and circles (open triangle and circles positioned throughout the margin) are named support vectors. These elements are the ones that determine the classifier, and so, if they were in a different position, the margin and the hyperplane position would not be the same anymore, and consequently, the classifier developed would be distinct. This logic is symbolic of the special way of functioning of support vector machines algorithms when compared to other machine learning methods in the way that it shows that the SVM models are more concerned on the observations of different classes that are more similar to each other (the ones closer to the margin) rather than focusing on the extreme examples of each class (the ones farther away of the margin). The right panel of figure 8 (Kirchner & Signorino, 2018) demonstrates a more realistic scenario where it is clear the pertinence of using the soft margin concept. It is not uncommon to find data that is not completely separable and so the idea of the soft margin is to allow some misclassification errors and to find a hyperplane that almost separates the two classes. The generalization of the maximum margin classifier to the non-separable data scenarios it is called support vector classifier (Haltuf, 2014). In this case, the support vectors are not only the vectors that lie directly on the margin but also the ones that lie on the incorrect side of the margin. The degree to which misclassification errors are allowed is defined in the C tuning parameter. The value for C is usually fixed via cross-validation, and the greater the C, the more tolerant the model will be with infringements to the margin, and by consequence the margin will also become larger. With that expansion, a bigger number of support vectors will be determining the hyperplane since there are more violations to the margin. In contrast, with a lower C, few support vectors will define the hyperplane, and one can expect a classifier with lower bias but higher variance as well (James et al., 2013).

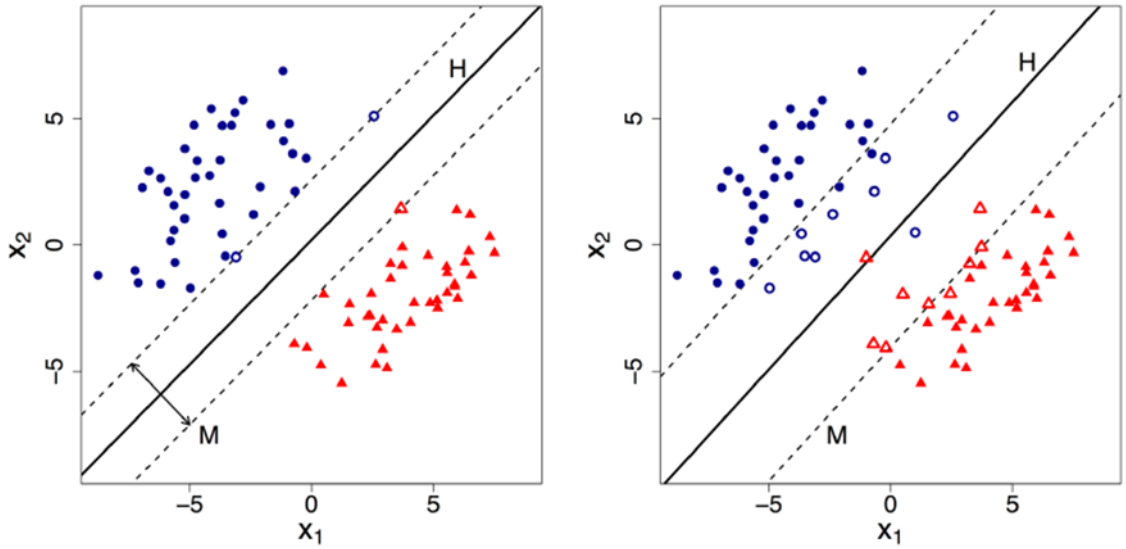

Figure 8 - Linear SVM Classification (Kirchner & Signorino, 2018)

When one faces a nonlinear scenario like the one shown in figure 9, the support vector classifier will not be adequate to handle the classification problem. The target variable is not linear in relation to X_1 and X_2 predictive variables and, thus, no straight line can be drawn to separate the two classes successfully without incurring in an unreasonable number of misclassifications. This is when support vector machines come in handy. Support vector machine is an extension of support vector classifier enlarging the feature space using the kernel tricks (James et al., 2013). This rationale is in line with the Cover's theorem which states that one can, with high probability, transform a non-linearly separable dataset into a linearly separable one projecting it into a space of higher dimension (figure 9). In a similar line of thought, a high dimensionality feature space is more likely to be linearly separable than a low dimension one, given that space is not densely populated (Cover, 1965). With the use of the nonlinear kernel functions the explicit calculation of the transformed vectors to achieve higher dimensionality is no longer needed and so the high computational effort associated with it is avoided. The kernel functions consist of measures of similarity between observations which correspond to the dot product of those observations in a higher-dimensional space, i.e. the dot product of the transformed vectors (P.Scheunders, D.Tuia, & G.Moser, 2018). In fact, the kernel functions take as input the observations in the original input space and return their similarity in the projected, higher dimensional, space. The kernel function also named simply kernels,

$$k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R} \tag{3}$$

can be defined as

$$k(x, x') = \langle \varphi(x), \varphi(x') \rangle_v \tag{4}$$

where $\langle \cdot, \cdot \rangle_v$ is the dot product in the Hilbert space and φ the mapping function.

As a key restriction, $\langle \cdot, \cdot \rangle_v$ must be a proper inner product. In fact, as long as $\mathcal{V}$ consist of an inner product space, the explicit representation of φ is not needed (P.Scheunders et al., 2018). With the kernel trick, a linear support vector classifier is found in the expanded space (right panel of Figure 10) and then it maps to a nonlinear classifier in the original space (S.Wilks, 2019). Figure 10, shows the result of a trained SVM with a nonlinear classification boundary, and the margin associated to it, in the

original data space (left panel) which has been obtained from the linear classifier visible in the right panel (S.Wilks, 2019). The kernel trick is not exclusive of support vector machines' algorithms. In fact, any algorithm that depends solely on the dot product of the observations can integrate this adaptation (Haltuf, 2014).

Although there are several kernel functions, a popular one is the radial basis function which is represented by the following equation:

$$k(x, x') = exp\ (-\gamma \parallel x - x' \parallel^2) \tag{5}$$

Where $\parallel x - x' \parallel^2$ is the squared Euclidean distance between two samples represented in two feature vectors and γ is the inverse of the standard deviation of the radial basis function kernel. The gamma value (γ) of the radial kernel works as a parameter (for example, in the function of SVM classifier from Scikit-learn's library) that defines the distance a single training observation can reach (Deepajothi & Selvarajan, 2013). Low values on the gamma parameter mean that a single training sample will have a wider influence. Another parameter of the radial basis function kernel is C. This latter expresses the balance between training examples misclassifications against decision surface simplicity (Deepajothi & Selvarajan, 2013). Contrary to what is stated by (James et al., 2013), and at least in the function of SVM classifier available in Scikit-learn's library, low values on the C parameter will make the classifier more inclined to creating a smooth decision surface rather than classifying correctly every single training sample. This type of kernel searches for a support vector classifier in an infinite dimensionality space. This aspect of radial basis function makes this trick very flexible and able to fit a tremendous variety of decision boundary surfaces (Haltuf, 2014). Without the kernel trick, one would need to calculate the transformed vectors explicitly in the infinite-dimensional space, which would be impossible. Another characteristic of this kernel is its local behaviour, given that only nearby training observations have an effect on the class label of a new observation (James et al., 2013). Because of this characteristic, one can say that this type of kernel has a style of classification that is close to another machine learning algorithm, which is k-nearest neighbours.

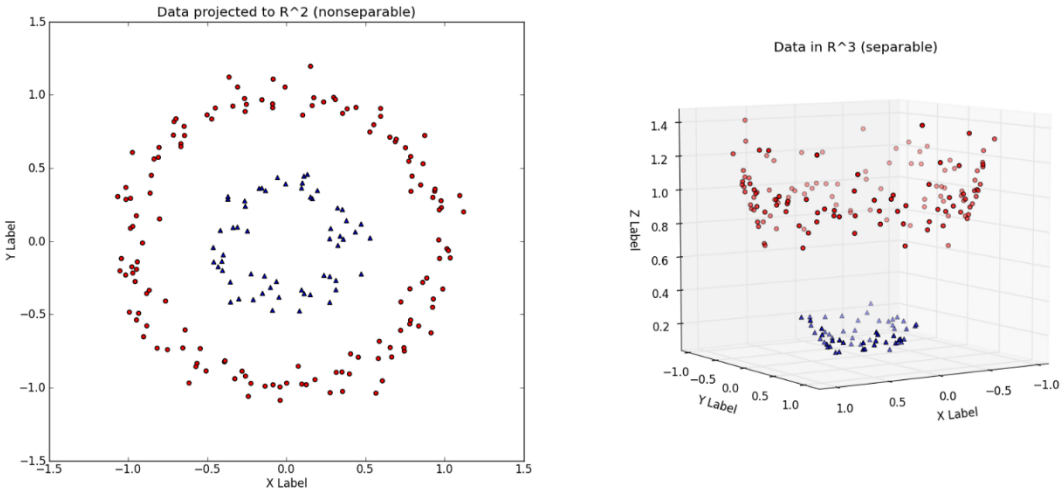

Figure 9 - Projection of the original input space in a higher dimension space

From https://towardsdatascience.com/understanding-the-kernel-trick-e0bc6112ef78

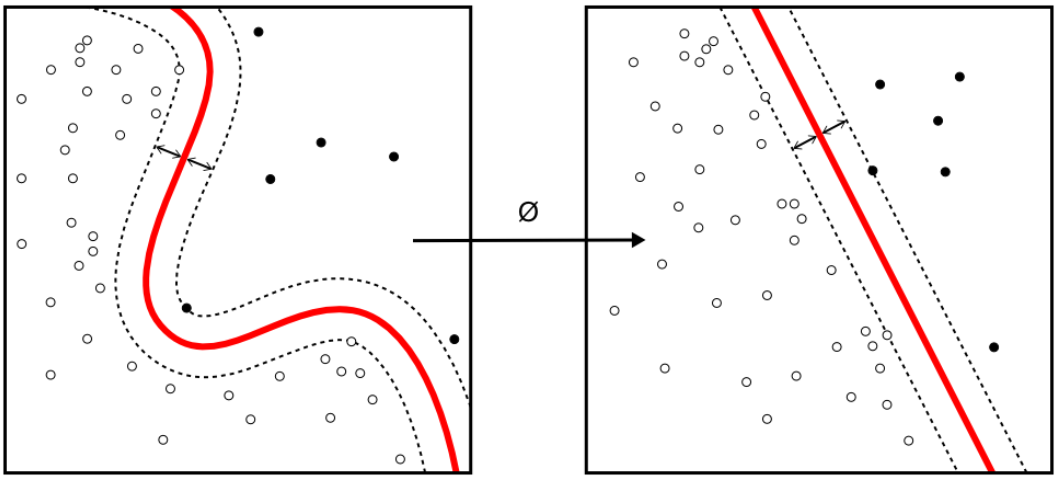

Figure 10 - Kernel Machine

From https://en.wikipedia.org/wiki/Support-vector_machine#/media/File:Kernel_Machine.svg

3.4.1.3. Decision tree

Although the concept of decision tree is not recent, decision tree algorithms have been gaining popularity with the growth of machine learning. This technique uses mathematical formulas like Gini index to find an attribute of the data, and also a threshold value of that attribute, in order to make splits of the input space (Patil & Sareen, 2016). In classification problems, those splits are performed in a way that best partitions the data in terms of discriminating individuals according to the class they belong to. Applied to the context of credit scoring, the algorithm will find the variables and the optimal split rules to end up, in each final bucket, with a group of individuals that are similar between them in terms of probability of default. In fact, besides the idea of grouping together individuals with similar probability of default, one wants at the same time to form groups that are different between them (Anderson, 2007). When it comes to understanding the splitting criterion, one must be familiar with the concepts of entropy and information gain. Entropy determines the impurity of a collection of individuals (Molala, 2019) in terms of their class. Moreover, it shows in which extent we are facing a group of individuals that is balanced in terms of the number of default and non-default (the two possible classes of our dependent variable). On the other hand, the concept of information gain refers to the expected reduction in entropy caused by partitioning the individuals using a given variable.

Regarding entropy, as referred by (Akanbi, Oluwatobi Ayodeji Amiri, Iraj Sadegh Fazeldehkordi, 2015), there are two different concepts associated with the logic of operation of the algorithm.

- **Target entropy:** one uses all the data (or all the data in the parent node if the first split was already done) to count how many individuals there are in each class and consequently calculate the entropy

$$Entropy\ (Set) = -\sum_{i=1}^{k} P(value_i).\log_2(P(value_i)) \quad (6)$$

being K the number of classes of the dependent variable

- **Feature's entropy:** for each feature, we calculate the expected child entropy, which is given by the entropies of the children weighted by the proportion of total examples of the parent node that are in each child.

$$Entropy\ (Feature\ A) = \sum_{i \in Values(A)} \left(\frac{Nº\ \ of\ instances\ in\ child\ node\ _i}{Nº\ of\ instances\ in\ the\ parent\ node}\right). Entropy(child\ node\ _i) \quad (7)$$

being A a given feature of the dataset and Values(A) the possible values of that feature.

The information gain we get from splitting with the several feature options is assessed, and the feature that maximizes the information gain is the chosen one to do the split. The tree grows split after split and, in the end, in classification problems like credit scoring, it classifies the instances considering the most frequent class in each leaf. In case of regression problems, entropy indicator is not adequate to measure the impurity, so this is done calculating the weighted mean squared error of the children nodes. In the end, the predicted value of the target variable corresponds to the average of the true values of the samples in each node.

Decision trees result in models with low bias and are commonly associated with overfitting. That aspect is regarded as one of the major disadvantages of this technique. One popular way to control it is pruning the tree fixing a limit for its depth. In fact, trees with many levels and with terminal nodes that contain only a few numbers of samples is a typical overfitting scenario. On the other hand, there are advantages associated with decision trees models, for instance, the results easy to interpret and a clear capacity for modelling features that are not necessarily linear.

Concerning the parameters of this machine learning algorithm, Max_depth is a popular one that exists embedded in the model available in Scikit-learn library, and that represents the depth (or the number of levels) of the tree. As long one keeps increasing this parameter, one will get closer from a scenario where the classifier is able to classify all of the training data correctly. From a certain level onwards, it is better to stop since overfitting will certainly emerge. One can also use two parameters that control splitting by establishing minimum numbers of samples that have to be respected in order to perform more splits. Specifically, Min_Samples_Split expresses a minimum number of samples that an internal node needs to contain in order to do a new split whereas Min_Samples_Leaf is a minimum number of samples required to be at the leaf nodes or, also called, terminal nodes (Fraj, 2017).

3.4.2. Ensemble classifiers

One of decision trees models main disadvantages is that they are prone to overfitting due to its high variance and dependence on the training data. Although it is possible to control the decision tree fitting process in order to stop overfitting another valid alternative to avoid overfitting that exists in machine learning sphere, are the adoption of ensemble methods. Ensemble methods, in opposition to single learning approaches, train several base learners. This type of learning became popular in the 1990s, where two pioneering work were conducted. One, from Hansen and Salamon (1990), demonstrated that predictions resulting from a combination of several classifiers were more accurate than the ones performed by a single classifier. The other one, with Schapire (1990), demonstrated that weak learners (learners that perform just slightly better than random) could be turned into strong learners (Chapman, 2014).

One way to segment the ensemble methods is according to the moment of generation of the several base learners. They can be generated in a sequential way, i.e. one after the other, or in a parallel process, i.e. the base learners are generated in parallel (Xia, Liu, Li, & Liu, 2017) (figure 11). The first segment is also called boosting, and these methods aim to lower bias and achieve a low generalization error. The second group of methods, also called bagging, tries to reduce variance but leaving at the same time bias unaffected (Vaishnavi, 2017).

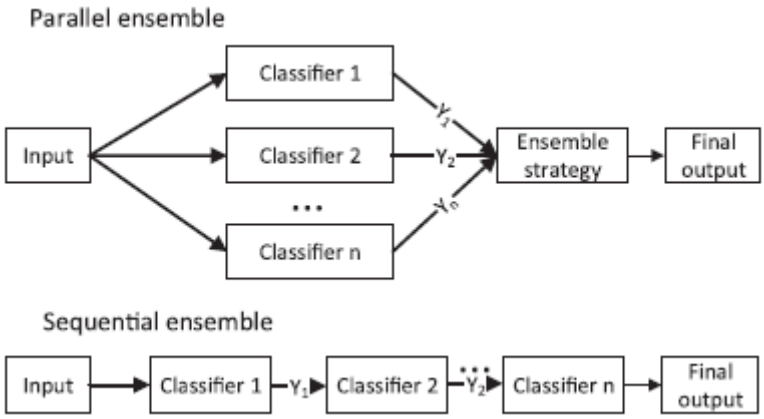

Figure 11 - Parallel and sequential ensemble flux *(Xia et al., 2017)*

3.4.2.1. Random forest

Random forest is part of the second group, and it is a method capable of performing both classification and regression problems, that uses decision trees as base learners. It was introduced by Breiman in the early 2000s and became very popular since then. This approach relates to the concept of bagging, which implies the generation of different training sets for the various individual learners. In fact, the model is trained with samples drawn randomly with replacement (bootstrap samples) from the training dataset and using random feature selection in the process of each single tree generation (Brown & Mues, 2012). Therefore, in the case of having k trees (base learners), k samples are formed in a uniform way, collecting, with replacement, Z observations from the training set. The randomness associated with the way the training occurs in random forest algorithm contributes to the generated individual trees to be non-correlated between them.

Although random forests are used both for classification and regression problems, in the context of credit scoring it is used as an ensemble classifier. In case of classification, the algorithm chooses the class with the majority vote (figure 12) while in case of regression it averages the several outputs of the different models (Bhatia, Sharma, Burman, Hazari, & Hande, 2017). The underlying logic is that considering a set of individual learners to form a prediction gives more accurate results than relying on the prediction of a single learner. Random forest is seen as one of the most efficient machine learning algorithms since it is comparably insensitive to skewed distributions, outliers and missing values (Carvajal, Marko, & Cullick, 2018). Moreover, the predictive variables for random forest method can be of any type: numerical, categorical, continuous or discrete (Carvajal et al., 2018).

This machine learning algorithm has the same parameters options described earlier for decision tree algorithm, but it also has another popular one called N_estimators (Scikit-learn library's terminology) that specifies the number of base learners being trained. Usually, the higher the number of estimators, the higher capacity of the model for learning with the training data (Fraj, 2017). Above a certain threshold that depends on the data and problem we are solving, there is no advantage in increasing this parameter as the computation effort will be much superior, and the performance on the test set can even decrease.

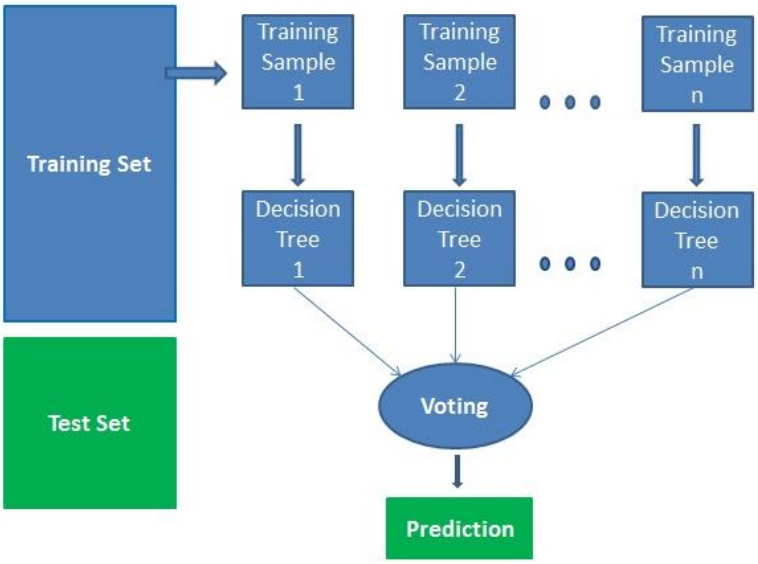

Figure 12 - Random forest

From https://www.datacamp.com/community/tutorials/random-forests-classifier-python#how

3.4.2.2. Stacking with cross-validation

Among the ensemble techniques, stacking is another type of approach. Literature has been more focused on homogeneous ensembles, and the stacking technique considered in this study consists of a heterogeneous ensemble learning approach. It was first implemented by (Wolpert, 1992) with the designation of Stacked Generalization where it was explained that the method works by deducing the biases of the base learners with respect to the training set. The deduction proceeds by building a (second-level) model that takes as inputs the outputs of the original base learners (first-level models). Base learners that are trained with a part of the training set and that make predictions, after the training, using the rest of the training set (Wolpert, 1992).

The Mlxtend library, Python library for machine learning tasks, firstly made available a stacking algorithm called StackingClassifier. This technique follows the generic spirit of stacking algorithms in the way it combines multiple classification models (first level classifiers) via a meta-classifier (second level classifier). Given a certain training dataset, all that data is used for training the base learners. Consequently, the outputs of those first models on the data used for training serve as features to the second level classifier fitting. As features for the second-level classifier can be used either probabilities or the predicted class (figure 13) (Raschka, 2014a). Stacking is a general framework in the way one can use single or ensemble learning approaches to generate first or second level classifiers. Compared with

other ensemble approaches such as bagging or boosting, stacking will work on discovering the optimal way of combining the base classifiers while the other two approaches will use voting (Aggarwal, 2014).

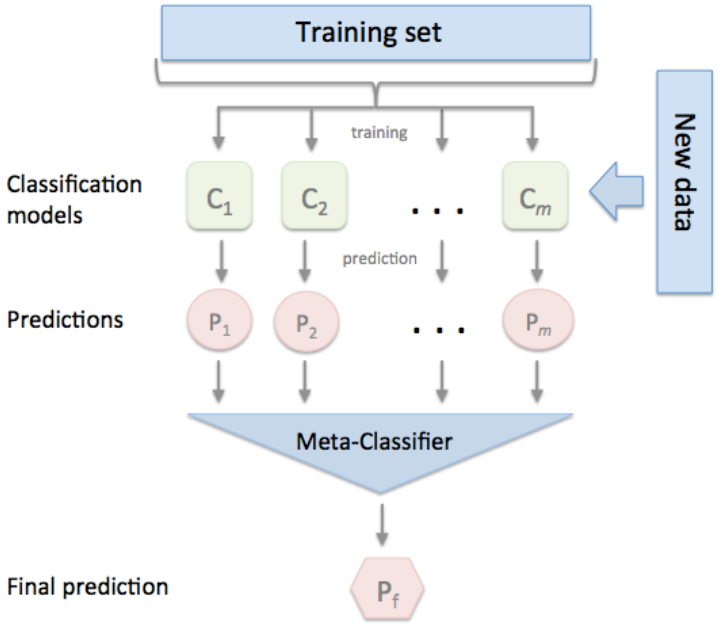

Figure 13 - Stacking classifier (Mlxtend library)

From http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/

The Mlxtend library's stacking algorithm presents the drawback of being prone to overfit. That is essentially because the features used in the second level model are derived from the same data that was used to train the first-level models (Aggarwal, 2014; Raschka, 2014a). To overcome this disadvantage, there is an alternative algorithm in the same Python Library (Mlxtend) called StackingCVClassifier. This alternative implements a similar logic but with the novelty of implementing the concept of cross-validation on the preparation of the input data for the second-level classifier. K-Fold is the applied cross-validation method, and it is also the most frequently used one to assess classification performance (Aggarwal, 2014; Raschka, 2014a).

The StackingCVClassifier algorithm (figure 14) starts by splitting the training dataset into k different folds. The number of k will dictate the number of iterations that will be performed. In each iteration, one-fold is left out while the others are used for training the first-level learners (our base learners) (Raschka, 2014b). Once those first-level learners are fitted, the algorithm uses those trained models to predict probabilities using as input the data contained in the previously left out fold (data that was not seen by the model while fitting). The process is repeated for each of k iterations, and the resulting probabilities are stacked and, in the end, used as input data to train our second-level learner (meta classifier) (Raschka, 2014b). After that, the algorithm repeats the first-level learners training process, this time using the whole training set. Applying the second level classifier on this updated first-level classifiers' output, results in the final ensemble model output (Aggarwal, 2014).

The other four, previously described, classifiers of this study were used as first-level classifiers for the stacking algorithm. The previously tuned hyperparameters of those classifiers were considered in this

first-level classifiers' training. Logistic regression model was used as second-level classifier, and a new hyperparameter tuning was performed giving the fact that new data (first-level classifiers' outputs instead of the original dataset features) was being inputted to this logistic regression model.

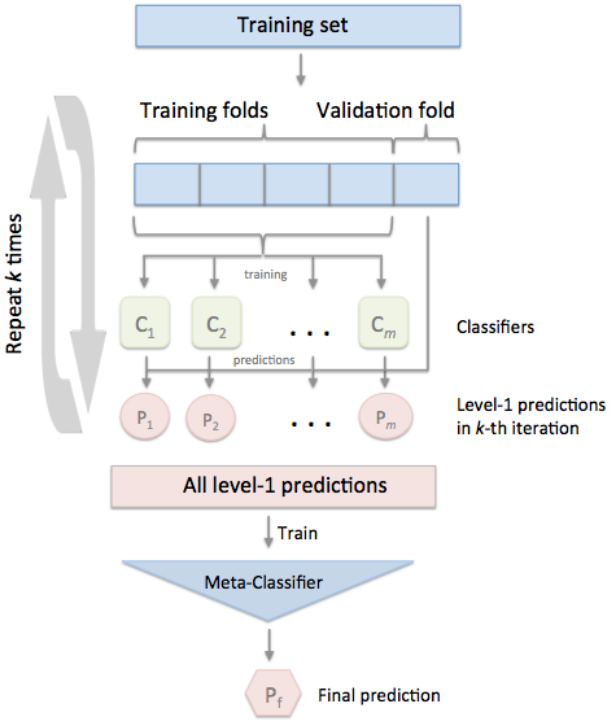

Figure 14 - Stacking with cross-validation classifier (Mlxtend library)

From http://rasbt.github.io/mlxtend/user_guide/classifier/StackingCVClassifier/

3.5. MODELLING TECHNIQUES

3.5.1. Sample split

In order to control overfitting and evaluate the model's capacity for generalization, it is necessary to keep a subset of the data out of the model training process. Only by doing that, one can be sure of the model effective performance and capacity for predicting accurately with new observations that were not "seen" during the training phase of the algorithm. As so, each of the four datasets used was randomly partitioned into two parts in a stratified way using Scikit-learn train_test_split function which means the proportion between default and non-default cases is equal in both training and test set. The percentage allocated for training was 70% while the remaining part was allocated for testing. Although there is not an explicit validation set, parameter tuning was performed using only the training set and through a five-fold cross-validation approach like the one illustrated in figure 15. More concretely, for every combination of hyperparameters, the model was trained five times (using in each time a different allocation of the training set between training and validation), and the performance was measured in those five iterations using the left-out fold destinated for validation in each iteration.

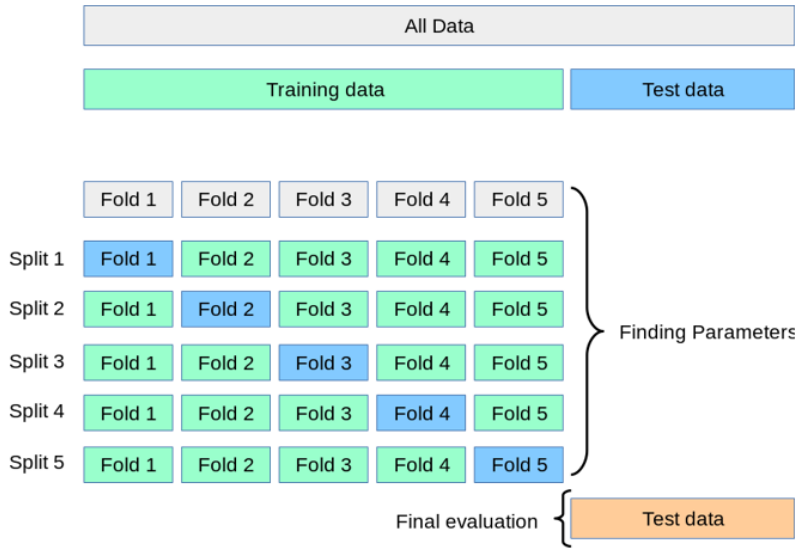

Figure 15 - Train and test split method and cross-validation

From https://amueller.github.io/ml-training-intro/slides/03-cross-validation-grid-search.html#8

3.5.2. Class reweight

Two of the four datasets used in this dissertation are strongly imbalanced (the more drastic case is the Kaggle dataset with only 7% of default records). Imbalanced datasets is a common scenario in machine learning classification problems where one can easily find almost all instances being classified with the most frequent label while far fewer being classified with the other label, which is generally the most important one (Kotsiantis, Kanellopoulos, & Pintelas, 2006). An obvious example of that is the training of a classifier on a fraud detection or credit scoring problem, where there will be an inclination for the classifier to classify all instances as "non-fraud". This issue is of major importance since, usually, a false negative cost has a much higher cost when compared to a false positive cost (Minh, 2018). In the specific context of credit scoring, losing one good client due to a false positive prediction means that the company will lose, essentially, the opportunity of receiving the loan's interests. On the other hand, accepting a bad client (false negative case) means that the company would lose, in the worst-case scenario, all borrowed amount.

To deal with this inconvenient, resampling strategies like over-sampling and under-sampling are common approaches, but that are not free from important critics. The first one, over-sampling, consists of increasing the minority class instances in the training set replicating records of that class. The second one, under-sampling, consists of an opposite approach of eliminating instances of the majority class (figure 16). If in one hand over-sampling is said to be prone to overfitting, on the other hand, under-sampling has the risk of losing useful information associated to it (Minh, 2018) and both methods have the inconvenient of making the features appear to have a different variance from the variance they truly do have (Fawcett, 2016).

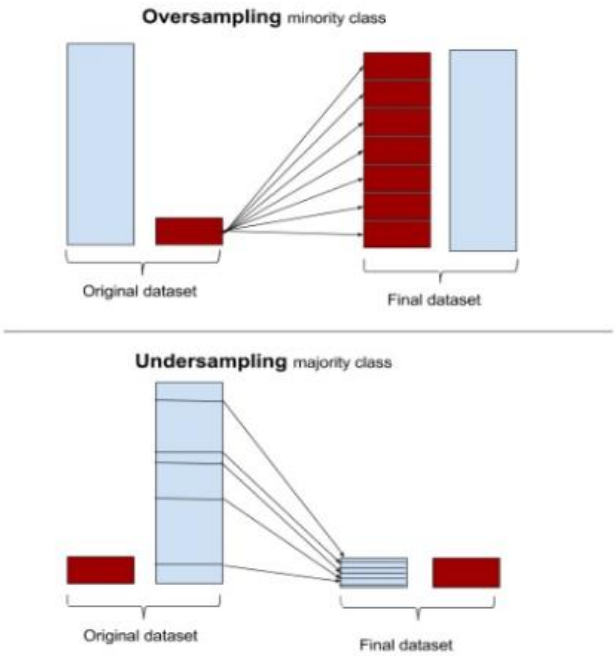

Figure 16 - Over-sampling and Under-sampling techniques

From https://www.svds.com/learning-imbalanced-classes/

An alternative strategy was followed in this study to make classifiers sensitive to the class imbalance underlying the several datasets. It was used a class weighted learning strategy where different weights are assigned to the two classes in order to the errors of the minority class to turn out more costly and the classifier, for that reason, will try harder to correctly classify them while fitting the data. The parameter class_weight set to "balanced" at the algorithm level function served this purpose and has strong advantages like its simplicity and common application in the credit risk context (Minh, 2018). This "balanced" option, as stated in Scikit-learn documentation, automatically adjusts weights inversely proportional to class frequencies, as

$$n\_samples / (n\_classes * np.bincount(y)) \quad (8)$$

The parameter class_weight set to "balanced" was used in every model developed and, in every dataset, where they were developed, including the datasets that were not strongly imbalanced, namely the Japanese and the Australian.

3.5.3. Hyperparameters optimization

Hyperparameters optimization was performed using a random search approach with the RandomizedSearchCV function from Scikit-learn. This method was shown in several empirical studies to be more efficient than the traditional method used for searching the optimal set of hyperparameters named grid search (Ju, Yang, Yang, Gegov, & Zhou, 2019). While grid search does an exhaustive search over all specified possible hyperparameter combinations, randomized search does a more efficient search. In fact, random search tries stochastically different values within the search space (Ju et al., 2019), to look for an optimal parameter combination over a pre-specified number of iterations. The number of iterations picked for this study was 30, and the search was performed exclusively using the

training set with a five-fold cross-validation approach. The performance obtained with each combination of hyperparameters was measured using the AUC metric. The pre-defined set of hyperparameters values given as input to randomized search algorithm are the ones stated below (table 7-11). The hyperparameters that were finally used in the models are the ones stated in the same tables (in "Tune Hyperparameters" columns) and correspond to the randomized search output, with an exception for the three models that are marked with *, in these cases parameters were defined manually.

Logistic Regression		Tuned Hyperparameters			
Hyperparameter	Set of Values	Kaggle	Japanese	German	Australian
penalty	l1, l2	l1	l1	l2	l1
C	1.0, 2.8, 7.7, 21.5, 59.9, 166, 464, 1291, 3593, 10000	1.0	1.0	2.8	2.8

Table 7 - Hyperparameters (Logistic regression)

SVM (RBF kernel)		Tuned Hyperparameters			
Hyperparameter	Set of Values	Kaggle*	Japanese	German	Australian
C	0.001, 0.01, 0.1, 1, 10	0.001	1.0	1.0	1.0
Gamma	0.001, 0.01, 0.1, 1	0.01	0.001	0.01	0.001

Table 8 - Hyperparameters (SVM with RBF kernel)

Decision Tree Classifier		Tuned Hyperparameters			
Hyperparameter	Set of Values	Kaggle	Japanese	German	Australian
max_depth	4, 5, 6, 7, 8	8	7	4	7
min_samples_leaf	5, 10, 15, 20, 50, 70, 100	70	50	50	50
min_samples_split	2, 5, 10, 15, 20, 50	5	50	5	50

Table 9 - Hyperparameters (Decision tree)

Random Forest Classifier		Tuned Hyperparameters			
Hyperparameter	Set of Values	Kaggle	Japanese*	German	Australian
max_depth	4, 5, 6, 7, 8	8	4	6	4
min_samples_leaf	5, 10, 15, 20, 50, 70, 100	50	27	15	10
min_samples_split	2, 5, 10, 15, 20, 50	20	10	5	50
n_estimators	10, 50, 100 , 200, 300	100	70	10	300

Table 10 - Hyperparameters (Random forest)

Stacking (Logistic Regression Meta-Classifier)		Tuned Hyperparameters			
Hyperparameter	Set of Values	Kaggle*	Japanese	German	Australian
penalty	l1, l2	l2	l1	l2	l2
C	1.0, 2.8, 7.7, 21.5, 59.9, 166, 464, 1291, 3593, 10000	1.0	7.7	1.0	2.8

Table 11 - Hyperparameters (Stacking with cross-validation)

3.6. PERFORMANCE EVALUATION

This section aims to describe the several methods that were used to assess the performance of the several models. The results of the methods described are presented only in the next chapter.

3.6.1. Confusion matrix and profit maximization

Confusion matrix is a popular performance measurement tool in classification problems (table 12). It consists of a contingency table that is useful to assess if the model is predicting each class in an accurate way and to calculate other performance metrics such as AUC. Associated with confusion matrix there is a commonly used terminology:

True Positives (TP) - positive instances that the model predicted correctly as positive

True Negatives (TN) - negative instances that the model predicted correctly as negative

False Positives (FP) - negative instances that the model predicted incorrectly as positive

False Negatives (FN) - positive instances that the model predicted incorrectly as negative

	Predicted **Non-Default** (Class 0)	Predicted **Default** (Class 1)
Actual **Non-Default** (Class 0)	True Negative No error	False Positive Type I Error
Actual **Default** (Class 1)	False Negative Type II Error	True Positive No error

Table 12 - Confusion Matrix structure

The confusion matrix was calculated for each developed model of each dataset used. To construct a confusion matrix there is one decision to be made: threshold definition. When one uses the Scikit-learn algorithm function named predict it is not uncommon, at least in probabilistic algorithms, that this function is assuming automatically a default threshold of 0.5 that, in result, will classify as positive every instance with probability (of being class 1) above 0.5 and will classify it as negative otherwise. So, if the confusion matrix is constructed using those calculated classes it will be constructed also with the underlying assumption of the default threshold of 0.5 (figure 17).

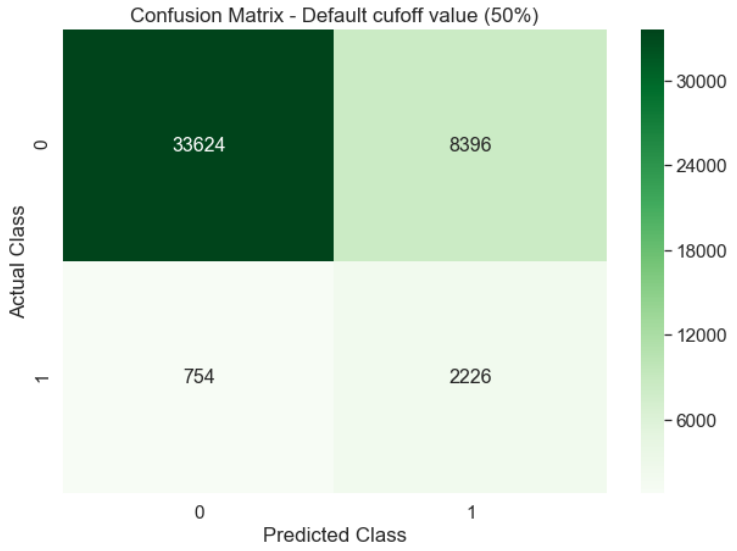

Figure 17 - Logistic regression confusion matrix (with a 0.5 threshold)

However, this default threshold of 0.5 in real life is not the most intelligent way to work with the developed model. One can, for example, gather the business context and produce a cost matrix that will help in determining the threshold that will maximize the profit (or minimize the cost) for the institution where the model is going to be applied. That cost matrix has the information of how much costs, for example, a false negative or how much money does the institution receives if it predicts a negative instance correctly as negative. For this dissertation it was, therefore, defined a fictional cost matrix that was considered in every model and in every dataset and that served as guiding information in the search for an optimum threshold (table 13). Given that cost matrix, the profit associated to each threshold was always calculated using the formula 9.

	Predicted **Non-Default** (Class 0)	Predicted **Default** (Class 1)
Actual **Non-Default** (Class 0)	+100	-100
Actual **Default** (Class 1)	-1 000	0

Table 13 - Cost matrix

$$Profit = 100 \times TN - 1000 \times FN \ - 100 \times FP \quad (9)$$

Note that since the business context of this classification problems is a credit scoring one, a more important cost (ten times higher) was set for Type II errors in comparison to Type I errors. According to (Kao, Chiu, & Chiu, 2012), the difference between these two types of costs, Type II and Type I, can range from 5 to 1 up to 20 to 1 respectively, since in the first case it is the issued amount that the institution is losing for lending money to a default borrower, while the second case it is just the amount

of interests that the institution did not earned because of the mistaken decision of rejecting a non-default client. As an example, figure 18 shows the profit curve of the logistic regression model developed in the Kaggle dataset. This curve shows the profit obtained for each threshold value and the point that maximized the profit was the one considered to reconstruct the confusion matrix as stated in figure 19. In a similar logic, the results chapter will state the several performance metric values considering always the optimum threshold in the calculation of those metrics instead of the default one of 0.5.

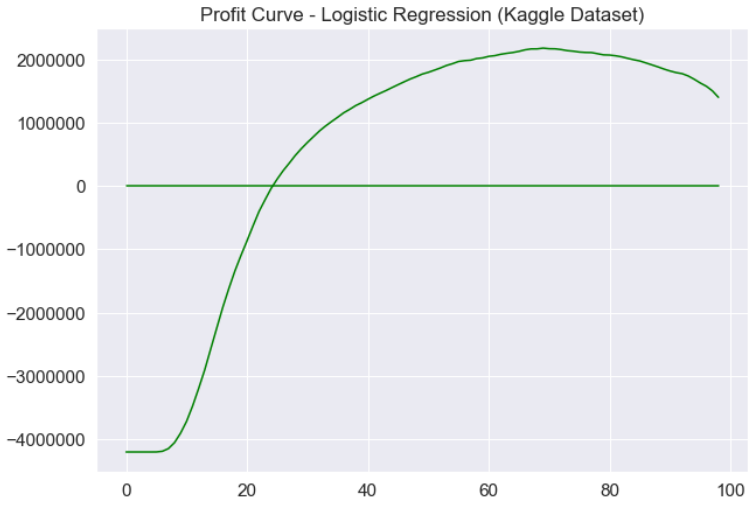

Figure 18 - Profit curve in logistic regression model (Kaggle dataset)

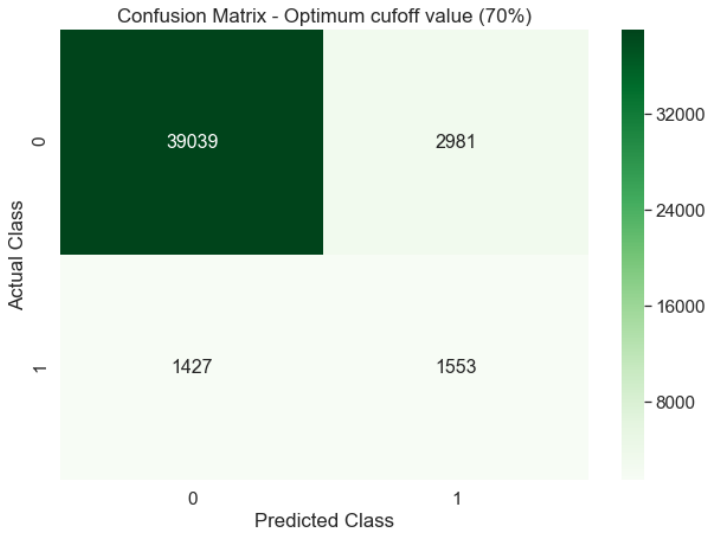

Figure 19 - Reconstructed confusion matrix (with the optimum threshold)

3.6.2. ROC curve – AUC and GINI index

One of the most popular and used metrics to assess model performance is the AUC metric. The AUC consists of the area under the ROC (Receiver Operating Characteristic) curve. The ROC curve (figure 20) is defined by the values of the true positive rate (also called sensitivity or recall) and the false positive rate (1 – specificity) at different decision thresholds values. While the first rate indicates the percentage of actual positives that the model classifies correctly, the second indicates the percentage of incorrect classifications over the actual negatives cases. If the model cannot accurately differentiate between positive and negative instances than the curve has a design that will be closer to the straight line between the lower left corner to the upper right corner. That line is the line associated with a model that makes random predictions. On the other hand, the more the curve bends towards the upper left corner, then the more capable is the model of differentiating between classes. Once we have the ROC curve, the performance of the model can also be expressed with the AUC value indicator (area under the curve). The higher the number, the more performant the model is.

Gini coefficient is also a popular performance indicator that is directly calculated through the AUC value according to the following formula:

$$Gini\ Coefficient = 2 \times AUC - 1 \tag{10}$$

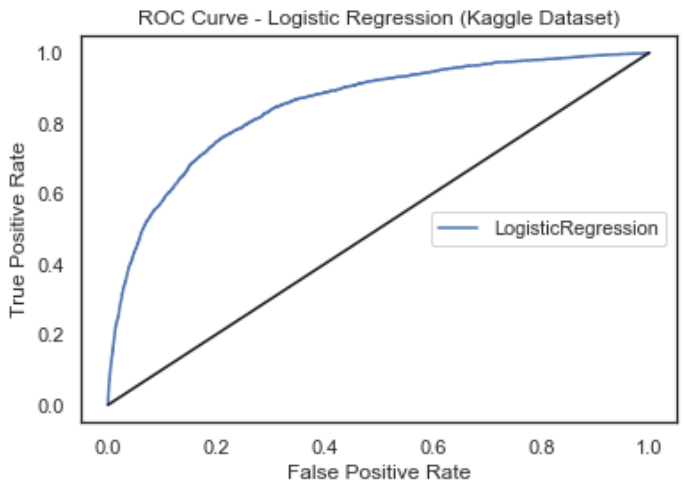

Figure 20 - ROC curve of logistic regression model in Kaggle dataset

3.6.3. Other performance indicators

To strengthen the performance evaluation analysis, several other indicators were calculated from the information contained in the confusion matrix. Those metrics enables the analyst to identify were the model is not being performant and allows a fair evaluation when facing imbalanced datasets.

$$Sensitivity, Recall\ or\ True\ Positive\ Rate = \frac{TP}{TP + FN} \tag{11}$$

$$Specificity, Selectivity\ or\ True\ Negative\ Rate = \frac{TN}{TN + FP} \tag{12}$$

$$Precision\ or\ Positive\ Predictive\ Value = \frac{TP}{TP + FP} \tag{13}$$

In binary classification problems, it is common to measure the model's Accuracy:

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \tag{14}$$

However, given the strong imbalance of two datasets of this study this indicator is not adequate since it can be truly misleading. For example, if a given model developed and tested in the Kaggle dataset turned out to be predicting non-default for all observations, that model would have an Accuracy value around 93%. This high value suggests that model is very performant when, in fact, is completely unable to identify a default case. For that reason, as an alternative, an indicator named Balanced Accuracy that is more appropriate to imbalanced datasets was calculated.

$$Balanced\ Accuracy = \frac{Sensitivy + Specificity}{2} \tag{15}$$

Another measure that is more appropriate for imbalanced datasets and useful when seeking balance between sensitivity and precision is the F1 Score which is the harmonic mean of both. In this particular context it was calculated a similar indicator (F-Beta Score) that has the convenient option of expressing different importance to sensitivity and precision. In this study, it was considered that sensitivity was ten times more important (given our cost matrix) than precision, so the beta considered was 10.

$$F_\beta\ Score = (1 + \beta^2) \cdot \frac{Precision\ \cdot Sensitivity}{(\beta^2 \cdot Precision) + Sensitivity} \tag{16}$$

4. RESULTS AND DISCUSSION

4.1. DETAILED RESULTS BY DATASET

The following tables (tables 14-17) show the values for several performance indicators calculated using the test set with an exception for the AUC Train Set indicator that is presented with the specific purpose of assessing the possibility of overfitting (by comparing it to the AUC indicator). There is one table for each dataset used, and each table has one column for each model developed in that particular dataset. All indicators reflect the defined threshold, i.e. the one that maximized the profit in Kaggle dataset and minimized the cost in the other three datasets. An exception to that is the AUC and Gini indicators, which are independent of the threshold choice.

Dataset: Kaggle	**Decision Tree**	**Logistic Regression**	**Random Forest**	**SVM** (RBF Kernel)	**Stacking** (with Cross Validation)
AUC Train Set	85,83%	84,86%	86,13%	84,86%	86,02%
AUC	84,76%	84,85%	85,36%	84,81%	85,38%
GINI	69,51%	69,71%	70,71%	69,62%	70,76%
Balanced Accuracy	73,23%	72,51%	73,87%	72,07%	73,32%
Sensitivity	54,56%	52,11%	56,11%	50,94%	54,40%
Specificity	91,90%	92,91%	91,64%	93,20%	92,24%
Precision	32,33%	34,25%	32,24%	34,70%	33,22%
F-Beta Score	54,19%	51,85%	55,70%	50,70%	54,05%
Profit/Cost	*2 167 200*	*2 178 800*	*2 191 200*	*2 168 600*	*2 191 200*

Table 14 - Results (Kaggle dataset)

Dataset: Japanese	**Decision Tree**	**Logistic Regression**	**Random Forest**	**SVM** (RBF Kernel)	**Stacking** (with Cross Validation)
AUC Train Set	92,79%	92,41%	93,25%	92,12%	93,06%
AUC	90,49%	91,46%	92,04%	91,58%	90,67%
GINI	80,97%	82,91%	84,09%	83,16%	81,34%
Balanced Accuracy	80,20%	76,47%	84,39%	74,35%	83,87%
Sensitivity	95,45%	97,27%	94,55%	98,18%	94,55%
Specificity	64,95%	55,67%	74,23%	50,52%	73,20%
Precision	75,54%	71,33%	80,62%	69,23%	80,00%
F-Beta Score	95,21%	96,92%	94,38%	97,78%	94,38%
Profit/Cost	*-2 100*	*-1 900*	*-1 300*	*-1 900*	*-1 500*

Table 15 - Results (Japanese dataset)

Dataset: German	Decision Tree	Logistic Regression	Random Forest	SVM (RBF Kernel)	Stacking (with Cross Validation)
AUC Train Set	75,05%	76,61%	78,68%	77,37%	76,73%
AUC	75,00%	80,09%	77,55%	79,76%	80,25%
GINI	49,99%	60,19%	55,10%	59,51%	60,49%
Balanced Accuracy	67,49%	72,68%	70,70%	71,18%	73,33%
Sensitivity	96,70%	95,60%	94,51%	94,51%	94,51%
Specificity	38,28%	49,76%	46,89%	47,85%	52,15%
Precision	40,55%	45,31%	43,65%	44,10%	46,24%
F-Beta Score	95,40%	94,57%	93,43%	93,45%	93,54%
Profit/Cost	*-7 900*	*-4 100*	*-6 300*	*-5 900*	*-4 100*

Table 16 - Results (German dataset)

Dataset: Australian	Decision Tree	Logistic Regression	Random Forest	SVM (RBF Kernel)	Stacking (with Cross Validation)
AUC Train Set	92,36%	92,05%	93,99%	91,73%	93,05%
AUC	91,44%	89,88%	92,68%	90,31%	91,75%
GINI	82,88%	79,77%	85,36%	80,63%	83,49%
Balanced Accuracy	72,27%	72,27%	71,83%	80,34%	70,19%
Sensitivity	97,62%	97,62%	99,21%	95,24%	98,41%
Specificity	46,91%	46,91%	44,44%	65,43%	41,98%
Precision	74,10%	74,10%	73,53%	81,08%	72,51%
F-Beta Score	97,31%	97,31%	98,86%	95,07%	98,07%
Profit/Cost	*-3 500*	*-3 500*	*-1 900*	*-3 500*	*-3 300*

Table 17 - Results (Australian dataset)

4.2. ROC CURVE – AUC ANALYSIS

To analyse the model capacity for separating the two classes of our classification problems, figures 21 - 24, display the several ROC curves constructed with the five models for each of the four datasets. The ROC curves plotted for the models developed in the Kaggle dataset reflect their similarity in terms of performance. In fact, all models scored an area under the curve of nearly 85% on the test set with the ensemble models (random forest and stacking) performing slightly better than the single classifiers.

In the Japanese dataset, the area under the curve of the several models was generically higher than the ones in Kaggle dataset, and the differences of performance between models showed to be a little more significant. The best model was random forest while the worst was the decision tree model, with a difference of 1.5% between the AUC of the two models.

The German dataset was the one where the developed models demonstrated to be less powerful in predicting the classes. However, the average AUC in this dataset was 78.5% which is still a good performance. Random forest and especially decision tree clearly underperform the other models, having their ROC curve lying below the other model's ROC curve for almost all threshold levels which means higher false positive and false negative rates on these cut-off points. Stacking with cross-validation was the model that showed superior AUC value having outperformed by 5.2% the decision tree model (the less performant model in this dataset).

Finally, in the Australian dataset, random forest outperforms all other models. Concretely, it demonstrates an AUC greater by 1% in relation to the second-best model (stacking with cross-validation) and greater by 2.8% in relation to the worst-performing model (logistic regression). In general terms, the models developed in this dataset along with the models developed in Japanese dataset showed a greater area under the curves, being the ROC curves closer to the top-left corner, which indicates a better capacity for separating default from non-default loans.

In conclusion, it is clear that the ensemble models performed generally better in terms of AUC in relation to other models, bringing better trade-offs between sensitivity and false positive rates. On average, across all datasets (figure 25), the best performance was the one from stacking algorithm (87%), being immediately followed by random forest algorithm (this latter with an AUC of 86.9%). Likewise, decision tree model showed a consistent weaker performance in relation to the rest of the models across the several datasets, having reached an average AUC value of 85.4%. The stacking and random forest algorithms were able to reach a slightly higher average AUC value in relation to the industry standard logistic regression model (greater by 0.4% and 0.3% respectively). The SVM classifier performed equally in relation to logistic regression (both models with an average AUC of 86.6%). However, the decision tree model underperformed, on average, the logistic regression model (-1.2% of AUC).

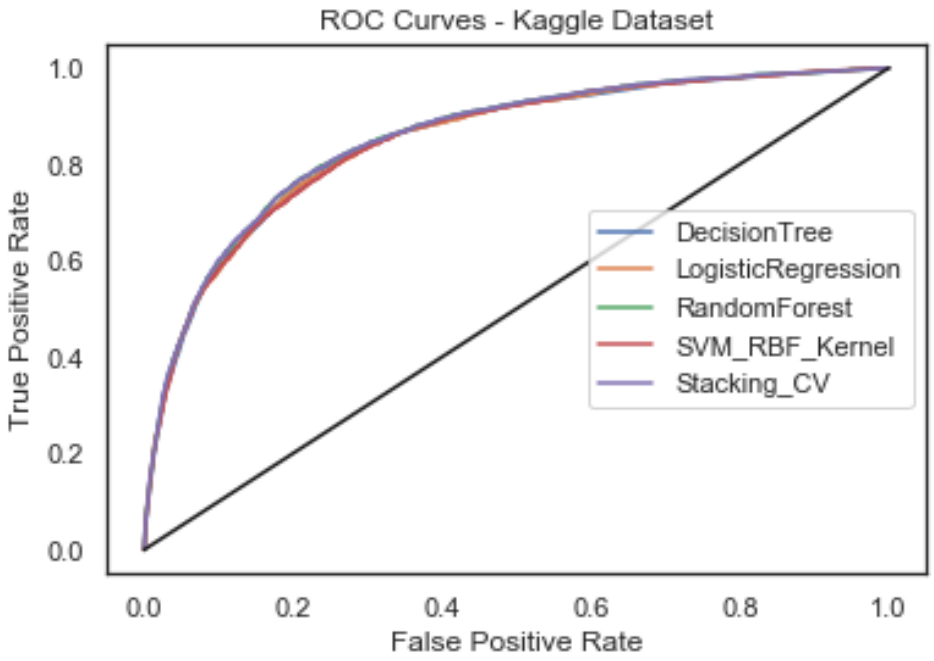

Figure 21 - ROC curves (Kaggle dataset)

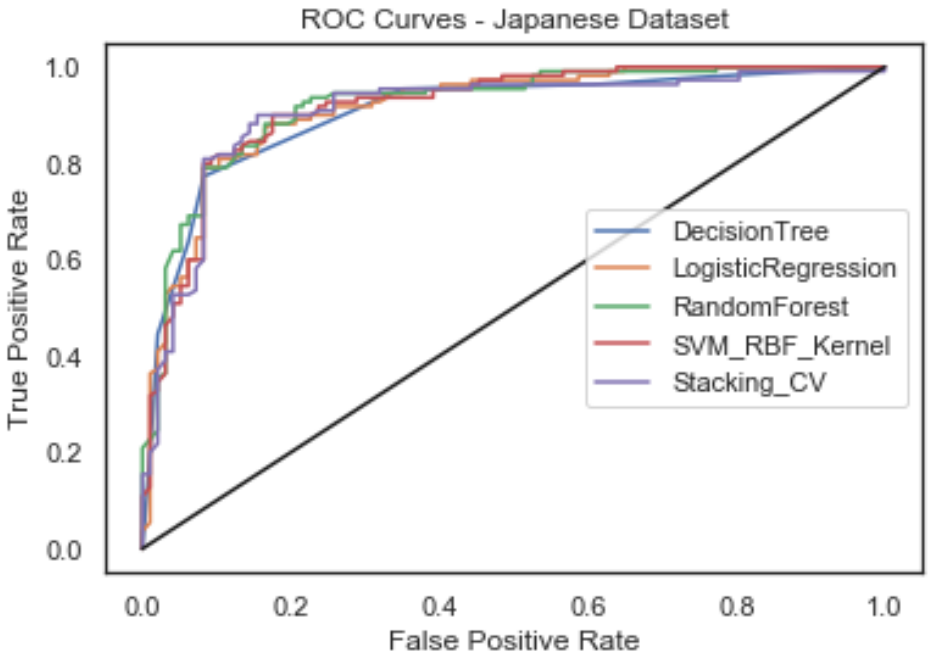

Figure 22 - ROC curves (Japanese dataset)

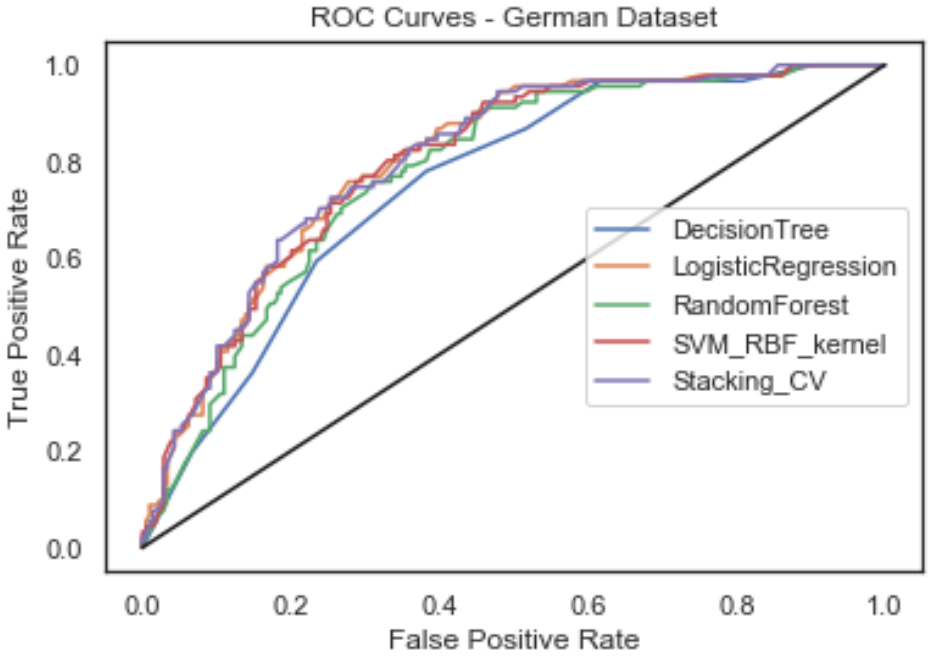

Figure 23 - ROC curves (German dataset)

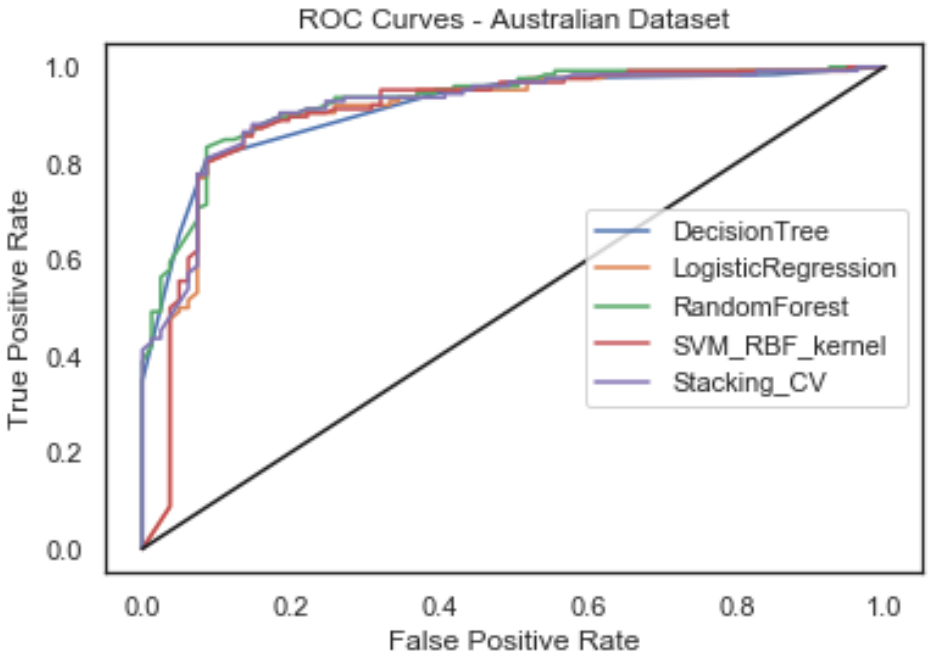

Figure 24 - ROC curves (Australian dataset)

4.3. AVERAGED INDICATORS ANALYSIS

The following graphs (figures 25 - 30) express the several performance indicators' average calculated on the values of the four datasets.

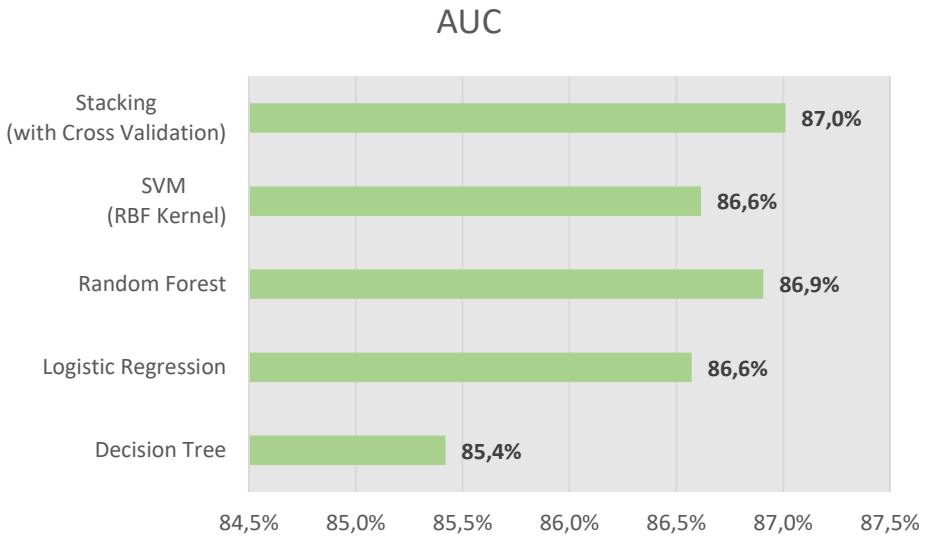

Figure 25 - AUC (average)

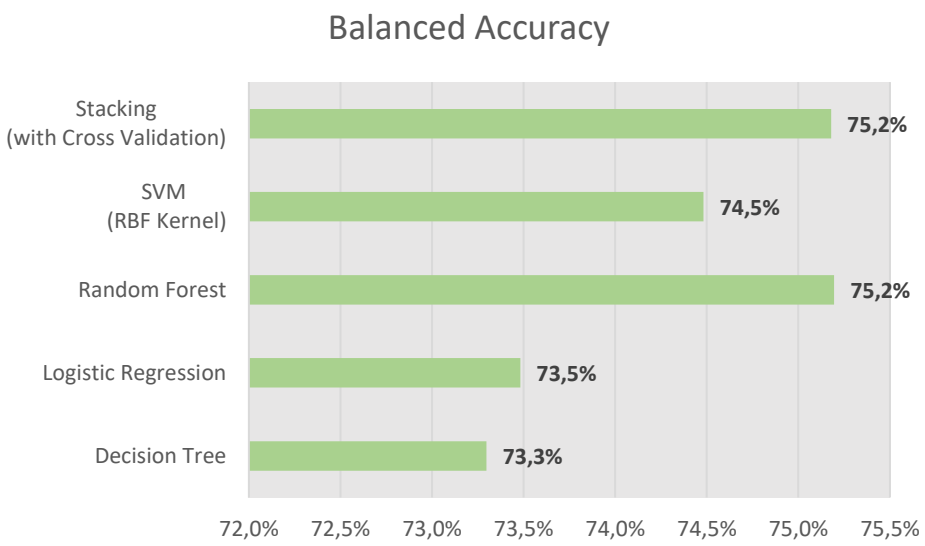

Figure 26 - Balanced Accuracy (average)

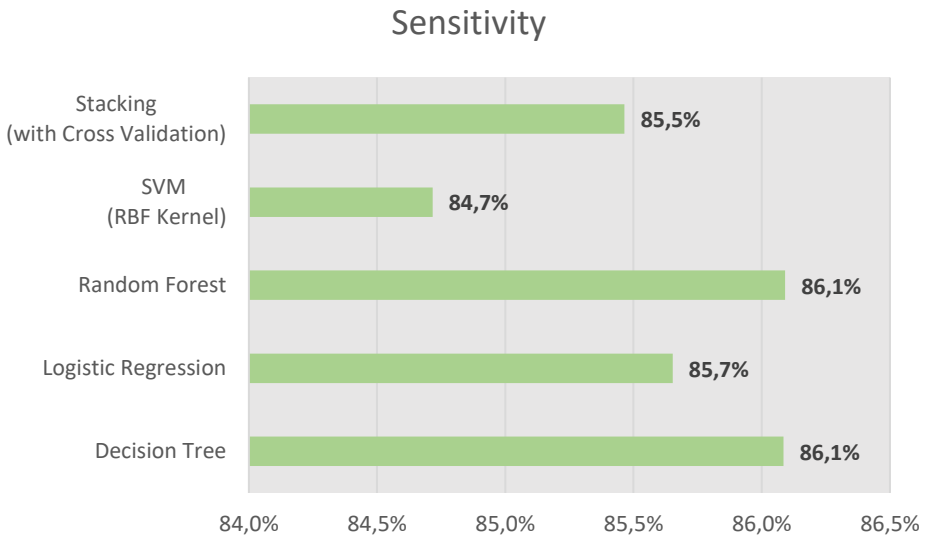

Figure 27 - Sensitivity (average)

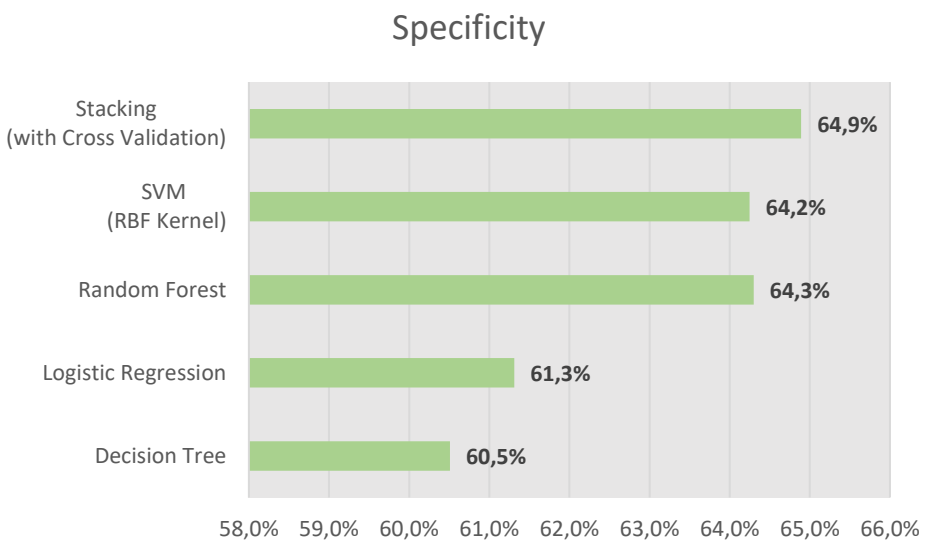

Figure 28 - Specificity (average)

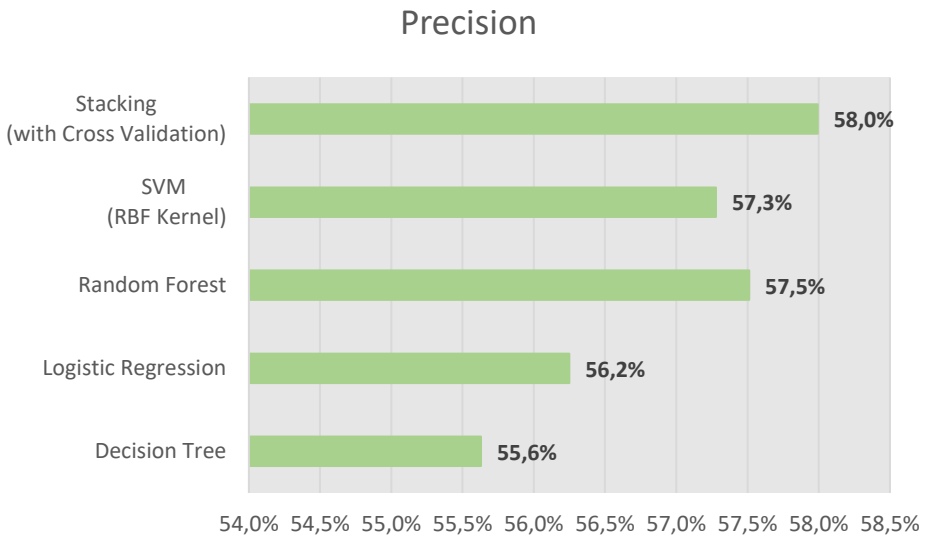

Figure 29 - Precision (average)

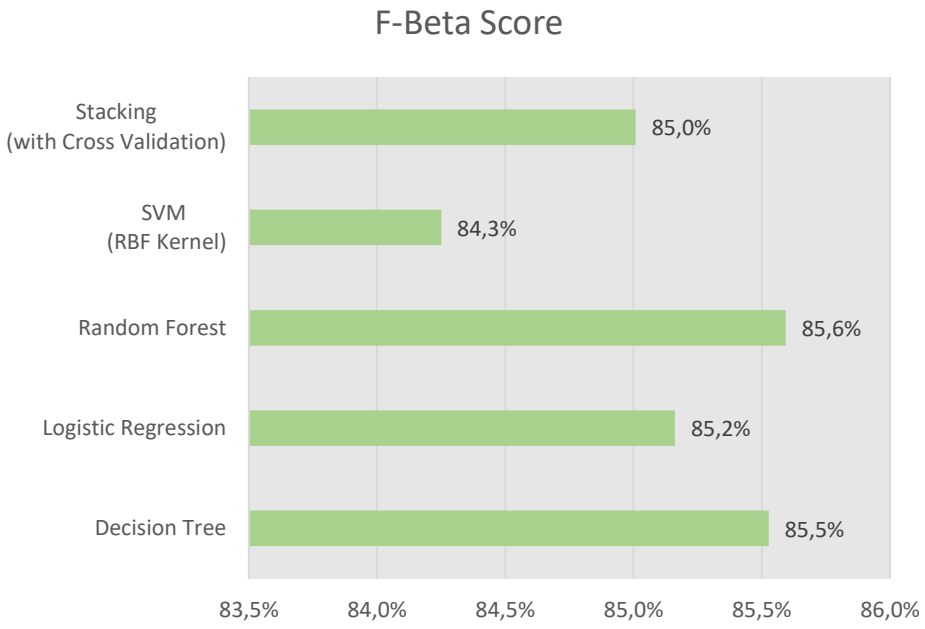

Figure 30 - F-Beta Score (average)

As this study was performed with imbalanced datasets (with two of them strongly imbalanced), for assessing accuracy the **balanced accuracy** indicator was preferred. In terms of this indicator, stacking and random forest showed an equivalent quality (75.2%). The worst performing model in terms of this indicator was decision tree model (73.3%). Not far from it was the logistic regression model (73.5%) which means that underperformed the two winning models in about 1.7%.

In terms of **sensitivity**, the tree learning approaches of this study (decision tree and random forest models) demonstrated the highest results (both with 86.1% of sensitivity). SVM model clearly underperformed in this indicator (84.7%) in relation to other models and logistic regression (85.7%) was only slightly worse in comparison to the winning models (less 0.4%).

In terms of **specificity** the average values were not that high except on the Kaggle dataset due to its sharp class imbalance. The winning model for this indicator was the stacking algorithm (with 64.9%). The losing one was the decision tree model (60.5%) but logistic regression was not much better (61.3%) having performed worse by 3.6% in relation to winning algorithm.

In terms of **precision**, the stacking model was the one with the highest value (58%). This model outperformed the worst model (decision tree model) in 2.4% and the industry standard logistic regression in 1.8% (this last registered a precision rate of 56.2%).

The **F-beta score** was calculated, as previously mentioned, with a beta equal to 10. The tree learning type models (decision tree and random forest) performed the best in this indicator, 85.5% and 85.6% respectively. Logistic regression demonstrated a slightly worse result, more concretely less 0.4% than the winning model (random forest).

4.4. FINANCIAL IMPACT ANALYSIS

The profitability of a credit scoring model is not an easy task to assess since it should be considered not only the ability to predict the probability of default but also the exposure at default (EAD) and the loss given default (LGD). Since this issue was not possible to address with this level of complexity, financial implications were measured using a simplified logic where the profit and costs of predicting correctly or incorrectly the several contracts were assumed to be the same for every instance (ignoring that different default cases can be associated with different loss amounts). In fact, it was used the cost matrix presented in the Performance evaluation section of the Experimental design chapter. The results are presented below.

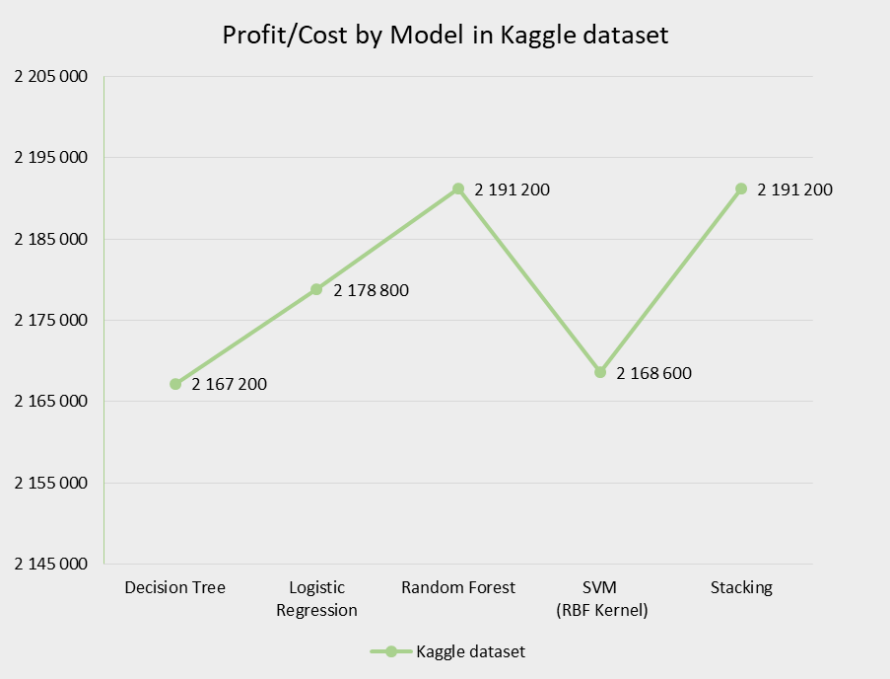

Figure 31 - Profit/cost by model (Kaggle dataset)

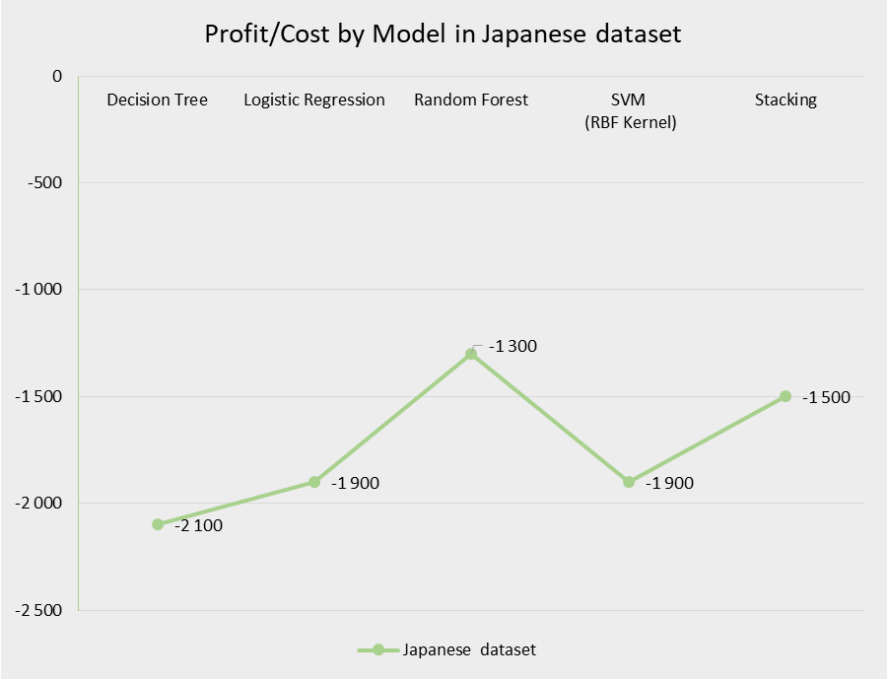

Figure 32 - Profit/cost by model (Japanese dataset)

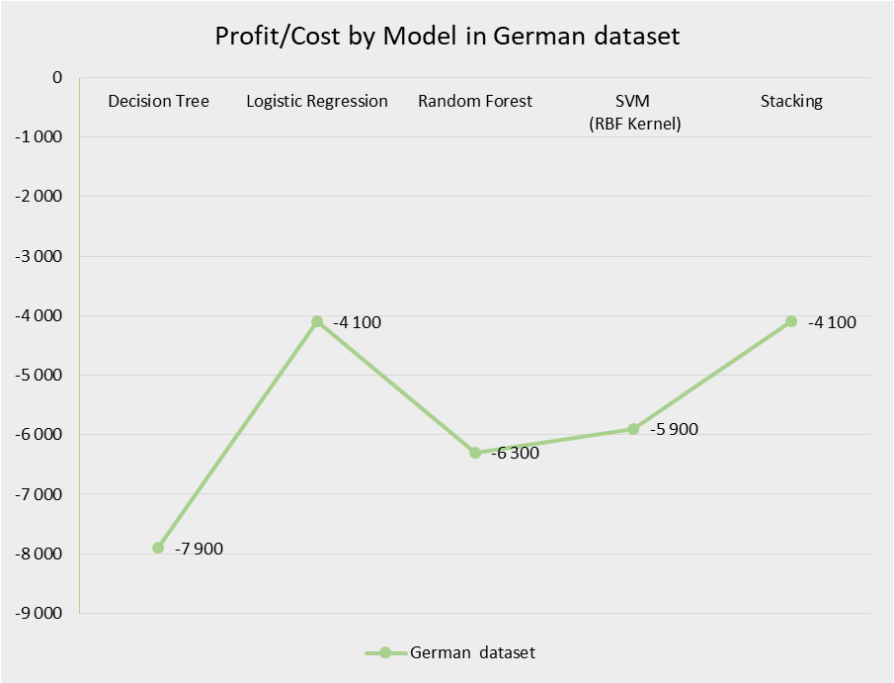

Figure 33 - Profit/cost by model (German dataset)

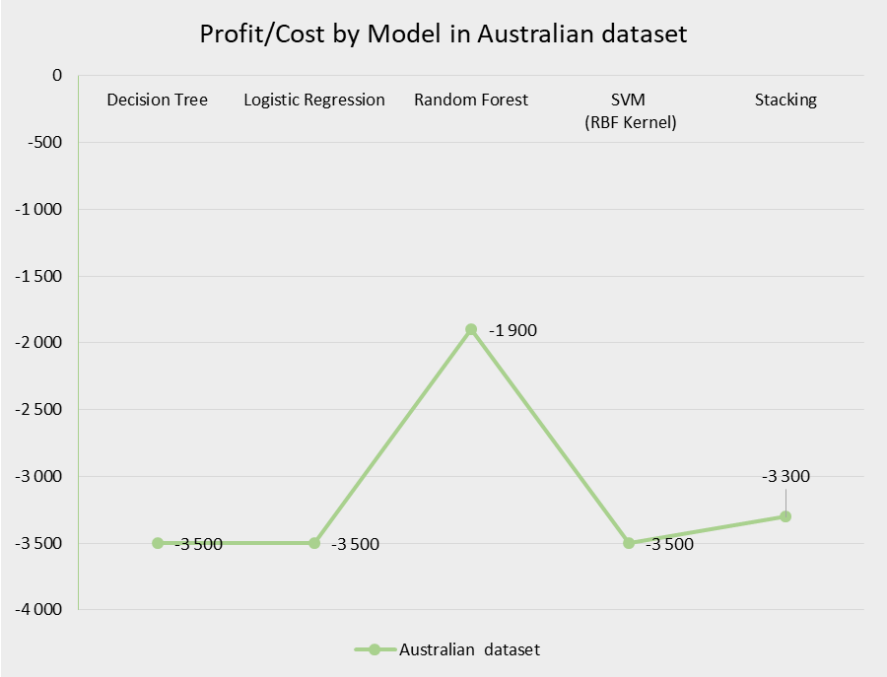

Figure 34 - Profit/cost by model (Australian dataset)

It is clear that the ensemble models (random forest and stacking) were able to bring, generically, a higher profit (or lower cost) across all datasets standing as first and second-best model. There was an exception to this which was the random forest in the German dataset (that was the second worst model) while the best models were the stacking and logistic regression (with an equivalent performance).

In relation to logistic regression, the stacking algorithm was able to originate a profit 0.6% higher in Kaggle dataset and was able to originate a cost 21.1% and 5.7% lower in the Japanese and Australian dataset respectively. In the German dataset the two models performed equally. Also, in relation to logistic regression, random forest was able to originate a profit 0.6% higher in the Kaggle dataset and originate a cost 31.6% and 45.7% lower in the Japanese and Australian dataset respectively. However, random forest originated a higher cost in comparison with logistic regression (in 53.7%) in the German dataset.

4.5. GLOBAL PERFORMANCE ASSESSMENT

To complement the described earlier analysis, four key performance indicators were selected, and the table below was produced. The table presents the number of models that perform worse than a given model (specified in the column header) on the selected indicators (specified in the rows) across the four datasets. The aim was to have a global score for each model that enables a quick and summarized comparison between them.

	Decision Tree	Logistic Regression	Random Forest	SVM (RBF Kernel)	Stacking (with Cross Validation)
Total Score	**18**	**28**	**43**	**22**	**41**
AUC	*2*	*7*	*12*	*7*	*12*
Balanced Accuracy	*6*	*7*	*10*	*6*	*10*
F-Beta Score	*10*	*8*	*9*	*5*	*7*
Profit/Cost	*0*	*6*	*12*	*4*	*12*

Table 18 - Results (global assessment)

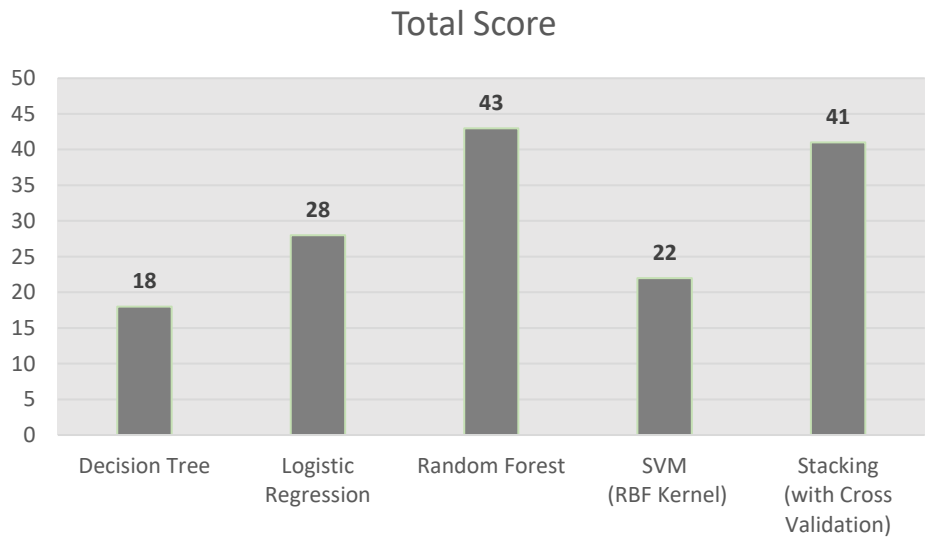

Figure 35 - Total score by model

This analysis reinforces the power of the ensemble methods mentioned in the previous points. Logistic regression appears in third, beating the other single classifiers in the group.

4.6. COMPARISON WITH RESULTS FROM PREVIOUS STUDIES

As mentioned earlier, this dissertation uses datasets that were also used in previous works. Therefore, it is possible to compare the results obtained in this dissertation with the ones obtained in other studies. (Moula, Guotai, & Abedin, 2017) established a comparison between six single classifiers constructed in six different datasets, of which four of them were the same datasets used in this study. Moreover, concerning the results of the six models in the four datasets that are common to this study, there is always a different winning model (in terms of AUC performance metric) for each of the four datasets. In the Australian, German and Kaggle dataset the winning models of this dissertation (random forest, stacking, and stacking respectively) outperform the winning models of (Moula et al., 2017) study (discriminant analysis, support vector machine and multi-layer perceptron) by 6.4%, 7.3% and 9.6%. However, regarding the Japanese dataset, the winning model of this dissertation (random

forest) underperforms the winning model of (Moula et al., 2017) study in 3.3% which consisted of a CART model.

Another study that presents an empirical evaluation with results that can be compared to the ones of this dissertation is the work from (Bequé & Lessmann, 2017). These authors developed eight individual classifiers over three different datasets (two of those datasets were also used in this dissertation) and developed as well, in those three datasets, more sixteen classifiers that consist of the application of two ensemble frameworks to the initial classifiers (eight models developed with bagging ensemble approach and another eight models with boosting approach). The datasets in common with this dissertation were the Australian and German datasets. Although the best performing model in this dissertation, in terms of AUC, for the Australian dataset (random forest) beats the identical approach of (Bequé & Lessmann, 2017) study (CART with bagging approach), it is also true that underperforms their winning model that consists of a logistic regression with a bagging approach by 1,7%. On the other hand, in the German dataset the winning model of this dissertation (stacking) performed equally to the winning model of (Bequé & Lessmann, 2017) that consisted of a support vector machine with radial basis function kernel. Moreover, both models reached an AUC of nearly 80.2%.

5. CONCLUSION

Credit scoring models' importance in banks and other financial institutions is clear, and it is undoubtfully pertinent to continue research for powerful methods in the machine learning sphere, so institutions have the best possible models to help them manage credit risk. The industry standard logistic regression was used in this study. Competing with it, there were two other single classifiers and two ensemble classifier methods (one of them heterogeneous, which is not frequently observed in credit scoring studies). Logistic regression showed to be not less performant than any of the other single classifier alternatives, and it has even outperformed decision tree classifier, having an average AUC greater by 1.2%. In the financial impact analysis, logistic regression was also never worse than its single classifiers non-parametric alternatives. On the other hand, the ensemble methods demonstrated better performance than logistic regression. Both ensemble methods applied in this study (random forest and stacking with cross-validation) reached, in comparison with logistic regression, a slightly higher AUC value (around +0.4%) as well as a higher balanced accuracy rate (+1.7%).

Comparing the homogeneous ensemble method (random forest) with the heterogeneous ensemble approach (stacking with cross-validation) the performance was not significantly different in terms of the averaged AUC value. Observing the detailed results by dataset, random forest outperformed stacking in AUC indicator in two of the datasets (Japanese and Australian) and performed equally on the Kaggle dataset. However, random forest did not succeed in the German dataset while stacking had the best performance much likely because it could lean on the logistic regression inputs (second-best performing model on this dataset). In general terms, random forest seem more capable of achieving the highest scores, but stacking has the advantage of being more capable of avoiding a significative bad performance due to its nature of combining different types of learning (for example if tree learning algorithms are not succeeding in a given task stacking will lean more on models of a different learning nature). Moreover, although in two of the four datasets, stacking algorithm managed to demonstrate AUC gains (reaching higher values of AUC compared to the ones of its base learners), those gains were not substantial. In fact, they were not strong enough to make the stacking algorithm be able to generate a higher profit (or a more reduced cost). In sum, both ensemble strategies outperformed the industry standard logistic regression. While the homogeneous approach is the main responsible for the highest performances, the heterogeneous approach demonstrates a good compromise between competitive performance and stability.

6. LIMITATIONS AND RECOMMENDATIONS FOR FUTURE WORKS

Further works that would enrich this analysis could be based in developing more stacking models using a different meta-classifier (in this study only logistic regression was used as meta-classifier) or even selecting different combinations of base learners to try to achieve better results. Including more ensemble techniques in the model comparison would also strengthen the analysis and underline their apparent superiority.

Moreover, the explainability of the ensemble models' results could be explored since it is a strong plus of the logistic regression method while being, commonly, an evident limitation of the ensemble methods. In fact, while this issue continues to be unsolved, the application of these more complex models will always be limited to business domains where the explainability is not crucial. Banks and other financial institutions will continue insisting that machine learning will have to come with explainability and so this is an important issue to keep exploring. More concretely, the Bank of England recently published a paper where the authors mention the inconvenient of machine learning black-box models and discuss, in particular, about the need for explainability in the specific field of default risk analysis. It mentions the increasing demand for model quality assurance and points explainable artificial intelligence as an important factor to achieve it (Bracke, Datta, Jung, & Sen, 2019). Also, the General Data Protection Regulation (GDPR) reinforces explainability as an obligation for businesses. In FICO blog, (Zoldi, 2017) makes it clearer pointing that article 22 of GDPR "*Automated individual decision-making, including profiling*" impacts on the subject of risk applications mentioning that customers need to have the reasons for how they were adversely impacted by a decision. In other words, it is mandatory to know what characteristics made the model reject the loan application. There are already some machine learning techniques to achieve this explainability and fight against the black-box curse. For instance, in the learning section of Kaggle website, a machine learning explainability subsection can be found. In specific, the SHAP Values technique helps in breaking down an individual prediction to find out the impact of each feature on it. That enables banks to know the reasons for which they are rejecting a loan while using their machine learning models. It is important to continue investing in explainable machine learning techniques to successfully benefit from all machine learning potential and decrease the banks skepticism related to it as well their dependence on the logistic regression method.

7. BIBLIOGRAPHIC REFERENCES

Abdou, H., Pointon, J., & El-Masry, A. (2008). Neural nets versus conventional techniques in credit scoring in Egyptian banking. *Expert Systems with Applications*, *35*(3), 1275–1292. https://doi.org/10.1016/j.eswa.2007.08.030

Abellán, J., & Mantas, C. J. (2014). Improving experimental studies about ensembles of classifiers for bankruptcy prediction and credit scoring. *Expert Systems with Applications*, *41*(8), 3825–3830. https://doi.org/10.1016/j.eswa.2013.12.003

Aggarwal, C. C. (2014). *Data ClassifiCation: Algorithms and Applications*. Chapman and Hall / CRC.

Akanbi, Oluwatobi Ayodeji Amiri, Iraj Sadegh Fazeldehkordi, E. (2015). *A Machine-Learning Approach to Phishing Detection and Defense*. Syngress.

Anderson, R. (2007). The Credit Scoring Toolkit - Theory and Practice for Retail Credit Risk Management and Decision Automation, 1–790.

Bach, M. P., Zoroja, J., Jakoviü, B., & Šarlija, N. (2017). Selection of Variables for Credit Risk Data Mining Models : Preliminary research, 1367–1372. https://doi.org/10.23919/MIPRO.2017.7973635

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., & Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society*, *54*((6)), 627–635.

Basel Committee on Banking Supervision. (2006). *International Convergence of Capital Measurement and Capital Standards*. *Bank for International Settlements*. Retrieved from http://www.bis.org/publ/bcbs128.pdf

Bequé, A., & Lessmann, S. (2017). Extreme learning machines for credit scoring: An empirical evaluation. *Expert Systems with Applications*, *86*, 42–53. https://doi.org/10.1016/j.eswa.2017.05.050

Boser, B., Guyon, I., & Vapnik, V. (1992). A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory – COLT '92* (p. 144).

Bracke, P., Datta, A., Jung, C., & Sen, S. (2019). Machine Learning Explainability in Finance: An Application to Default Risk Analysis. *SSRN Electronic Journal*, (816). https://doi.org/10.2139/ssrn.3435104

Brid, R. S. (2018). Logistic Regression [Blog post]. Retrieved from https://medium.com/greyatom/logistic-regression-89e496433063

Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, *39*(3), 3446–3453. https://doi.org/10.1016/j.eswa.2011.09.033

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. *Proceedings of the 23rd International Conference on Machine Learning*, *C*(1), 161–168. https://doi.org/10.1145/1143844.1143865

Carvajal, G., Marko, M., & Cullick, S. (2018). *Intelligent Digital Oil and Gas Fields*.

Castro Vieira, J. R. de, Barboza, F., Sobreiro, V. A., & Kimura, H. (2019). Machine learning models for credit analysis improvements: Predicting low-income families' default. *Applied Soft Computing*,

83, 105640. https://doi.org/10.1016/j.asoc.2019.105640

Chapman, H. (2014). *Ensemble Methods*. *Combining Pattern Classifiers*. https://doi.org/10.1002/9781118914564.ch6

Chopra, A., & Bhilare, P. (2018). Application of Ensemble Models in Credit Scoring Models. *Business Perspectives and Research*, *6*(2), 129–141. https://doi.org/10.1177/2278533718765531

Cibulskienė, D., & Rumbauskaitė, R. (2012). Credit Risk Management Models of Commercial Bankas: their Importance for Banking Activities. *Social Research*, *2*(27), 71–77.

Cover, T. M. (1965). Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *IEEE Transactions on Electronic Computers*, *EC-14*(3), 326–334. https://doi.org/10.1109/PGEC.1965.264137

Deepajothi, S., & Selvarajan, S. (2013). Performance Evaluation of SVM--RBF Kernel for Classifying ECoG Motor Imagery. *International Journal of Computer Science and Telecommunications*, *4*(5), 44–48.

Dietterich, T. G. (2000). An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees. *Machine Learning*, *40*, 139–157. https://doi.org/10.1023/A:1007607513941

Dzeroski, S., & Bernard, Z. (2011). Is Combining Classifiers Better Than Selecting The Best One, 1–8. Retrieved from papers://5e3e5e59-48a2-47c1-b6b1-a778137d3ec1/Paper/p1770

Fawcett, T. (2016). Learning from Imbalanced Classes [Blog post]. Retrieved from http://www.svds.com/learning-imbalanced-classes/

Ferreira, H. (2018). Dealing with categorical features in machine learning [Blog post]. Retrieved from https://medium.com/hugo-ferreiras-blog/dealing-with-categorical-features-in-machine-learning-1bb70f07262d

Finlay, S. (2011). Multiple classifier architectures and their application to credit risk assessment. *European Journal of Operational Research*, *210*(2), 368–378.

Fraj, M. Ben. (2017). In Depth: Parameter tuning for Random Forest [Blog post]. Retrieved from https://medium.com/all-things-ai/in-depth-parameter-tuning-for-random-forest-d67bb7e920d

Gahlaut, A., Tushar, & Singh, P. K. (2017). Prediction analysis of risky credit using Data mining classification models. *8th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2017*. https://doi.org/10.1109/ICCCNT.2017.8203982

Haltuf, M. (2014). University of Economics in Prague Faculty of Finance.

Hand, D. J., & Henley, W. E. (1997). Statistical classification methods in consumer credit scoring: A review. *Journal of the Royal Statistical Society. Series A: Statistics in Society*, *160*(3), 523–541. https://doi.org/10.1111/j.1467-985X.1997.00078.x

Harris, T. (2015). Credit scoring using the clustered support vector machine. *Expert Systems with Applications*, *42*(2), 741–750. https://doi.org/10.1016/j.eswa.2014.08.029

Hu, H., Li, J., Plank, A., Wang, H., & Daggard, G. (2006). A comparative study of classification methods for microarray data analysis. *Conferences in Research and Practice in Information Technology Series*, *61*(August 2014), 33–37.

Hue, S., Hurlin, C., & Tokpavi, S. (2017). Machine Learning for Credit Scoring: Improving Logistic Regression with Non Linear Decision Tree Effects, (July), 1–29.

Ince, H., & Aktan, B. (2009). A Comparison of Data Mining Techniques for Credit Scoring in Banking: a

Managerial Perspective. *Journal of Business Economics and Management Journal*, *10*(3), 233–240. https://doi.org/10.3846/1611-1699.2009.10.233-240

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning with Applications in R*. https://doi.org/10.2200/S00899ED1V01Y201902MAS024

Ju, Z., Yang, L., Yang, C., Gegov, A., & Zhou, D. (2019). *Advances in Computational Intelligence Systems: Contributions Presented at the 19th UK Workshop on computational Intelligence, September 4-6, 2019, Portsmouth, UK*. (Z. Ju, L. Yang, C. Yang, A. Gegov, & D. Zhou, Eds.). Springer.

Kao, L. J., Chiu, C. C., & Chiu, F. Y. (2012). A Bayesian latent variable model with classification and regression tree approach for behavior and credit scoring. *Knowledge-Based Systems*, *36*, 245–252. https://doi.org/10.1016/j.knosys.2012.07.004

Khandani, A. E., Kim, A. J., & Lo, A. W. (2010). Consumer Credit Risk Models via Machine-Learning Algorithms, 1–55.

Kirchner, A., & Signorino, C. S. (2018). Using Support Vector Machines for Survey Research. *Survey Practice*, *11*(1), 1–14. https://doi.org/10.29115/sp-2018-0001

Kotsiantis, S., Kanellopoulos, D., & Pintelas, P. (2006). Handling imbalanced datasets : A review. *Science*, *30*(1), 25–36. https://doi.org/10.1007/978-0-387-09823-4_45

Kruppa, J., Schwarz, A., Arminger, G., & Ziegler, A. (2013). Consumer credit risk: Individual probability estimates using machine learning. *Expert Systems with Applications*, *40*(13), 5125–5131. https://doi.org/10.1016/j.eswa.2013.03.019

L, V. K., Natarajan, S., Keerthana, S., Chinmayi, K. M., & Lakshmi, N. (2016). Credit Risk Analysis in Peer-to-Peer Lending System Vinod Kumar L Keerthana S , Chinmayi K M , Lakshmi. *2016 IEEE International Conference on Knowledge Engineering and Applications (ICKEA)*, 193–196. https://doi.org/10.1109/ICKEA.2016.7803017

Lawi, A., Aziz, F., & Syarif, S. (2018). Ensemble GradientBoost for increasing classification accuracy of credit scoring. *Proceedings of the 2017 4th International Conference on Computer Applications and Information Processing Technology, CAIPT 2017*, *2018–Janua*, 1–4. https://doi.org/10.1109/CAIPT.2017.8320700

Lee, T.-S., Chiu, C.-C., Chou, Y.-C., & Lu, C.-J. (2006). Mining the customer credit using classification and regression tree and multivariate adaptive regression splines. *Computational Statistics & Data Analysis*, *50*(4), 1113–1130. https://doi.org/10.1016/j.csda.2004.11.006

Lessmann, S., Baesens, B., Seow, H. V., & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *247*(1), 124–136. https://doi.org/10.1016/j.ejor.2015.05.030

Li, S., Ivor W., T., & Narendra S., C. (2012). Relevance vector machine based infinite decision agent ensemble learning for credit risk analysis. *Expert Systems with Applications*, *39*(5), 4947–4953.

Luo, C. (2018). A comparison analysis for credit scoring using bagging ensembles. *Expert Systems*, (January 2017), e12297. https://doi.org/10.1111/exsy.12297

MacKenzie, D. I., Nichols, J. D., Royle, J. A., Pollock, K. H., Bailey, L. L., & Hines, J. E. (2018). *Occupancy Estimation and Modeling (Second edition)*. Academic Press.

Marqués, A. I., García, V., & Sánchez, J. S. (2012). Two-level classifier ensembles for credit risk assessment. *Expert Systems with Applications*, *39*(12), 10916–10922. https://doi.org/10.1016/j.eswa.2012.03.033

Mester, L. J. (1997). What's the Point of Credit Scoring ? *Business Review*, *3*, 3–16. Retrieved from https://www.phil.frb.org/research-and-data/publications/business-review/1997/september-october/brso97lm.pdf

Minh, H. (2018). How to Handle Imbalanced Data in Classification Problems [Blog post]. Retrieved from https://medium.com/james-blogs/handling-imbalanced-data-in-classification-problems-7de598c1059f

Molala, R. (2019). Entropy, Information gain, and Gini Index; the crux of a Decision Tree [Blog Post]. Retrieved from https://blog.clairvoyantsoft.com/entropy-information-gain-and-gini-index-the-crux-of-a-decision-tree-99d0cdc699f4

Moula, F. E., Guotai, C., & Abedin, M. Z. (2017). Credit default prediction modeling: An application of support vector machine. *Risk Management*, *19*(2), 158–187. https://doi.org/10.1057/s41283-017-0016-x

P.Scheunders, D.Tuia, & G.Moser. (2018). Comprehensive Remote Sensing. *Reference Module in Earth Systems and Environmental Sciences*, *2*, 199–243.

Papouskova, M., & Hajek, P. (2019). Two-stage consumer credit risk modelling using heterogeneous ensemble learning. *Decision Support Systems*, *118*(October 2018), 33–45. https://doi.org/10.1016/j.dss.2019.01.002

Patil, P. S., & Sareen, V. (2016). An Overview of Classification Algorithms and Ensemble Methods in Personal Credit Scoring, *8491*, 183–188.

Qiu, G., Kandhai, D., & Sloot, P. M. A. (2010). Credit scorecard based on logistic regression with random coefficients. *Procedia Computer Science*, *1*(1), 2463–2468. https://doi.org/10.1016/j.procs.2010.04.278

Raschka, S. (2014a). StackingClassifier. Retrieved from http://rasbt.github.io/mlxtend/user_guide/classifier/StackingClassifier/

Raschka, S. (2014b). StackingCVClassifier. Retrieved from http://rasbt.github.io/mlxtend/user_guide/classifier/StackingCVClassifier/

S.Wilks, D. (2019). *Statistical Methods in the Atmospheric Sciences (Fourth Edition)*.

Teng, S., Du, H., Wu, N., Zhang, W., & Su, J. (2010). A cooperative network intrusion detection based on fuzzy SVMs. *Journal of Networks*, *5*(4), 475–483. https://doi.org/10.4304/jnw.5.4.475-483

Tsai, C.-F. (2014). Combining cluster analysis with classifier ensembles to predict financial distress. *Information Fusion*, *16*.

Vaishnavi. (2017). Ensemble Methods and Random Forests, (May), 1–9.

Vanderheyden, B., & Priestley, J. (2018). Logistic Ensemble Models. Retrieved from http://arxiv.org/abs/1806.04555

Wang, G., Hao, J., Ma, J., & Jiang, H. (2011). A comparative assessment of ensemble learning for credit scoring. *Expert Systems with Applications*, *38*(1), 223–230. https://doi.org/10.1016/j.eswa.2010.06.048

Wang, G., & Ma, J. (2012). A hybrid ensemble approach for enterprise credit risk assessment based on Support Vector Machine. *Expert Systems with Applications*, *39*(5), 5325–5331.

Wang, G., Ma, J., Huang, L., & Xuce, K. (2012). Two credit scoring models based on dual strategy ensemble trees. *Knowledge-Based Systems*, *26*, 61–68.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259. https://doi.org/10.1016/S0893-6080(05)80023-1

Xia, Y., Liu, C., Li, Y., & Liu, N. (2017). A boosted decision tree approach using Bayesian hyper-parameter optimization for credit scoring. *Expert Systems with Applications*, *78*, 225–241. https://doi.org/10.1016/j.eswa.2017.02.017

Zakirov, P. D., Baxter, J., Bartlett, P., & Frean, M. (2015). A Comparison of Data Mining Techniques in Evaluating Retail Credit Scoring Using R Programming. *2015 Twelve International Conference on Electronics Computer and Computation (ICECCO)*, 1–4. https://doi.org/10.1109/ICECCO.2015.7416867

Zhou, L., Lai, K. K., & Yu, L. (2010). Least squares support vector machines ensemble models for credit scoring. *Expert Systems with Applications*, *37*, 127–133.

Zhou, L., Tam, K. P., & Fujita, H. (2016). Predicting the listing status of Chinese listed companies with multi-class classification models. *Information Sciences: An International Journal*, *328*.

Zoldi, S. (2017). GDPR and Other Regulations Demand Explainable AI [Blog Post]. Retrieved from https://www.fico.com/blogs/gdpr-and-other-regulations-demand-explainable-ai